Encentuate® Identity and Access Management (IAM)

AccessStudio Guide

Product version 3.5

Document version 3.5.6

Contact information

For more information about this product or any support enquiries, contact us:

To log a support incident: http://support.encentuate.com/customerservice/

To reach us by phone:

- Singapore/Asia Pacific: +65-6471-1855
- USA: 1-800-ENCENTUATE

Table of Contents

About This Guide

Welcome to the Encentuate Identity and Access Management (IAM) AccessStudio Guide.

Use this guide to set up and maintain AccessProfiles in Encentuate IAM using AccessStudio.

Purpose

This guide provides procedures to setting up and maintaining AccessProfiles using Encentuate IAM's AccessStudio component.

Audience

This guide is meant for Administrators of Encentuate Identity Access Management (IAM). We assume that you have advanced computing knowledge, and are familiar with common computer and Windows-related terms.

What's in this guide

AccessStudio Installation details the instructions on installing and uninstalling AccessStudio.

About AccessStudio provides an overview of AccessStudio's features, benefits, functions, basic/advanced concepts, and interface.

Standard AccessProfiles provides an overview of AccessProfiles and details how to create, import, modify and save standard AccessProfiles using The AccessProfile Generator and Form Editor.

Advanced AccessProfiles provides an overview on how to create and edit advanced AccessProfiles using the Form Editor.

Managing Authentication Services provides an overview on how authentication services are associated with AccessProfiles. It also gives instructions on how to create, modify authentication services and manage authentication service groups and group links.

Managing Application Objects defines what an application object is and gives instructions on how to create and modify an application object.

Account Data Items and Templates details what an account data item and account data templates are and gives examples on how to view each type.

XPaths provides an overview on XPath, its supported axes, types, and available operators. It also lists XPath attributes for executables, windows, Web pages, HTML, and Java windows

Validating Functions details how to use the Validate XML Structure feature and check the task pane for any node errors.

Testing AccessProfiles describes the built-in real-time AccessStudio testing features and details instructions on testing AccessProfiles.

Downloading, Uploading, and Saving Information describes how you can download, upload and save AccessProfiles.

Backing up IMS Server Data provides steps on how to back up IMS server data.

Frequently Asked Questions provides a list of common AccessStudio user queries and answers.

Triggers lists the pre-defined triggers in AccessStudio.

Actions lists the pre-defined actions in AccessStudio.

Glossary defines all the commonly-used terms and abbreviations used throughout the guide.

Document conventions

Refer to this section to understand the distinctions of formatted content in this guide.

Main interface elements

The following are highlighted in bold text in the guide: dialog boxes, tabs, panels, fields, check boxes, radio buttons, fields, buttons, folder names, policy IDs/names, and keys. Examples are: **OK**, **Options** tab, and **Account Name** field.

Navigation

All content that helps users navigate around an interface is italicized (for example: *Start >> Run >> All Programs*)

Cross-references

Cross-references refer you to other topics in the guide that may provide additional information or reference. Cross-references are highlighted in green and display the referring topic's name (for example: Document conventions).

Hyperlinks

Hyperlinks refer you to external documents or Web pages that may provide additional information or reference. Hyperlinks are highlighted in blue and display the actual location of the external document or Web page (for example: http://www.encentuate.com).

Scripts, commands, and code

Scripts, commands, or codes are those entered within the system itself for configuration or setup purposes, and are usually formatted in a Courier font.

For example:

```
<script language="JavaScript">

<!--

  ht_basename = "index.php";

  ht_dirbase = "";

  ht_dirpath = "/" + ht_dirbase;

//-->

</script>
```

Tips or Hints

✓ *Tips or hints help explain useful information that would help perform certain tasks better.*

Warnings

! *Warnings highlight critical information that would affect the main functionalities of the system or any data-related issues.*

AccessStudio Installation

This chapter covers the following topics:

- Installing AccessStudio
- Uninstalling AccessStudio

Installing AccessStudio

Use the Install Encentuate AccessStudio wizard to install AccessStudio in your computer.

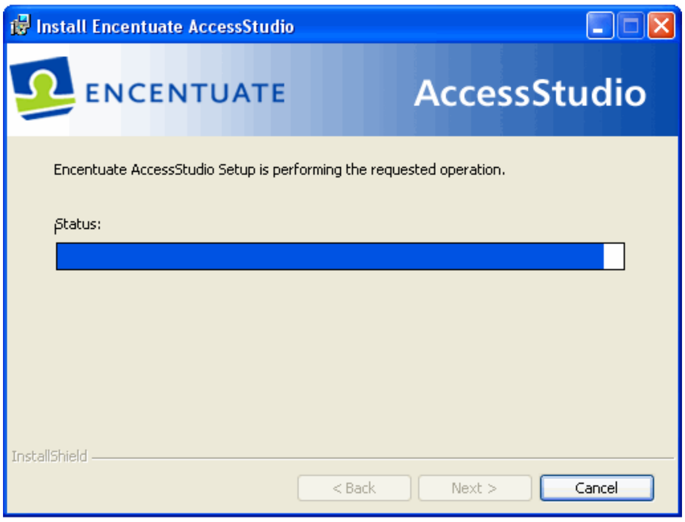
Before you install AccessStudio, ensure that you meet the following requirements:

- A compatible version of Encentuate AccessAgent pre-installed in your computer
- Microsoft.NET Framework 2.0 (http://www.microsoft.com/downloads/details.aspx?FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5&DisplayLang=en)
- At least an Intel® Pentium® III or equivalent processor
- A minimum of 256MB of RAM

To install AccessStudio:

Insert the Encentuate installation CD.

1. Go to *Start >> Run...*, click **Browse...**, and click **My Computer**. Right-click on the CD drive and select **Explore**.
1. Click on **AccessStudio.msi** icon in the Encentuate installation CD. The installation begins when the progress window appears.

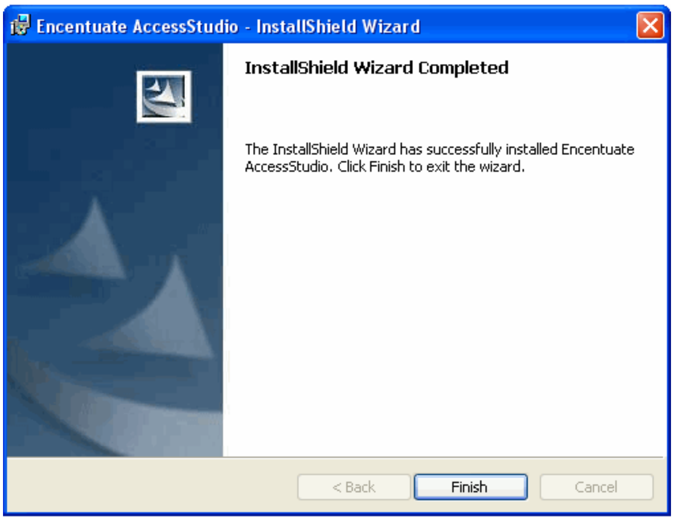


Encentuate AccessStudio Setup is performing the requested operation.

Status:

Installation progress bar

❷ Click **Finish** after a successful installation.



InstallShield Wizard Completed

The InstallShield Wizard has successfully installed Encentuate AccessStudio. Click Finish to exit the wizard.

Successful installation confirmation screen

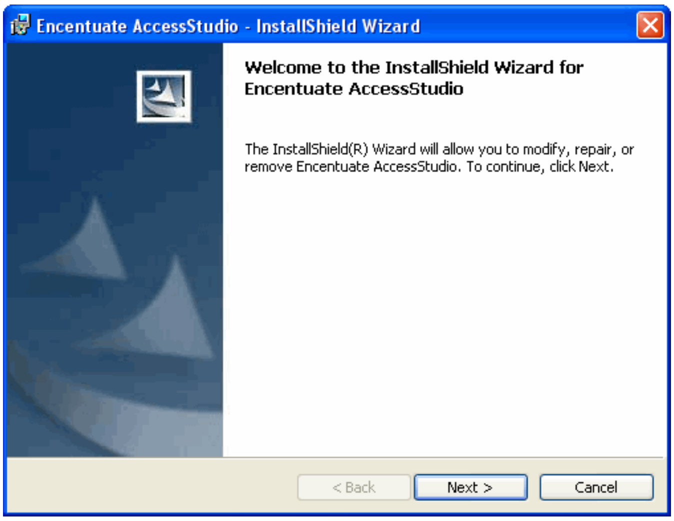
To launch AccessStudio, click *Start* >> *All Programs* >> *Encentuate Access-Studio* >> *AccessStudio*.

Uninstalling AccessStudio

Use the Install Encentuate AccessStudio wizard to remove the installation of AccessStudio in your computer.

To uninstall AccessStudio:

❶ Use the Windows Control panel to start the uninstallation process or launch the **AccessStudio.msi** icon from the Encentuate installation CD. The Encentuate AccessStudio - InstallShield Wizard appears.



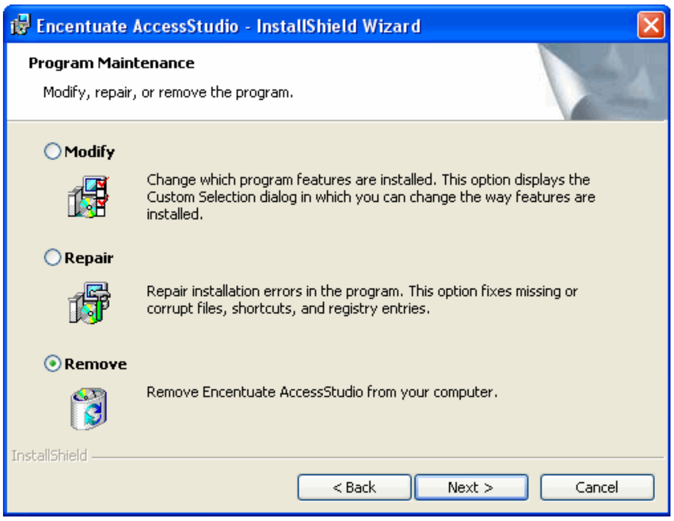
Encentuate AccessStudio - InstallShield Wizard

Welcome to the InstallShield Wizard for Encentuate AccessStudio

The InstallShield(R) Wizard will allow you to modify, repair, or remove Encentuate AccessStudio. To continue, click Next.

Click next to modify, repair or remove AccessStudio.

❷ Click **Next** to go to the next wizard window.



Encentuate AccessStudio - InstallShield Wizard

Program Maintenance

Modify, repair, or remove the program.

○ **Modify**

Change which program features are installed. This option displays the Custom Selection dialog in which you can change the way features are installed.

○ **Repair**

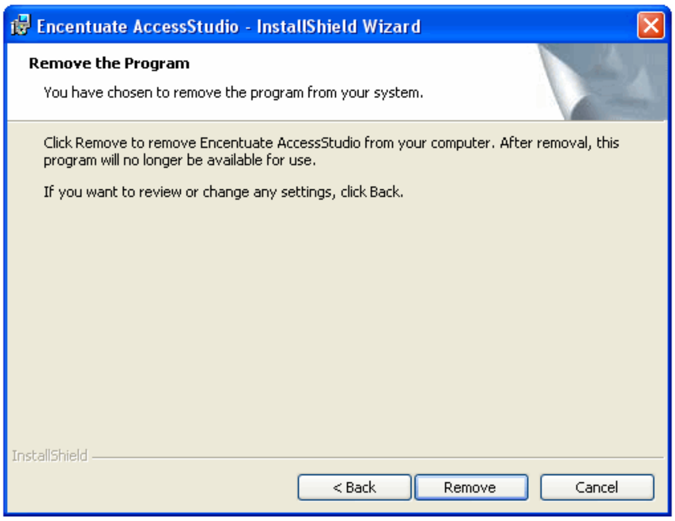
Repair installation errors in the program. This option fixes missing or corrupt files, shortcuts, and registry entries.

[✓ **Remove**]

Remove Encentuate AccessStudio from your computer.

Select the Remove radio button if you want to uninstall AccessStudio.

❸ To completely remove AccessStudio, mark the **Remove** radio button and click **Next**.


Encentuate AccessStudio - InstallShield Wizard

Remove the Program

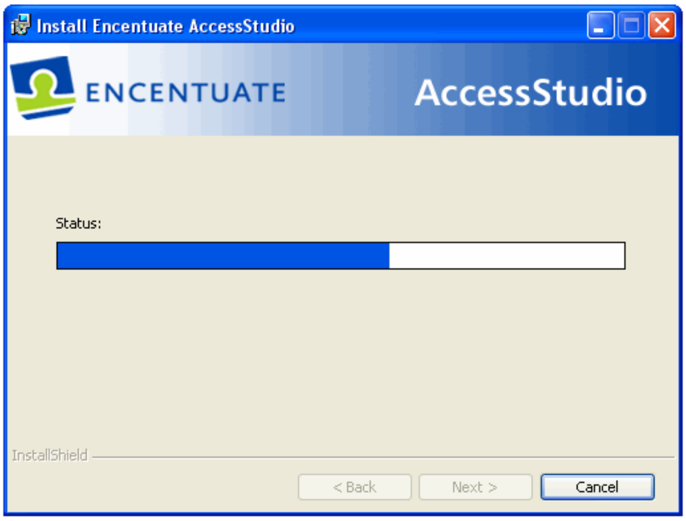
You have chosen to remove the program from your system.

Click Remove to remove Encentuate AccessStudio from your computer. After removal, this program will no longer be available for use.

If you want to review or change any settings, click Back.

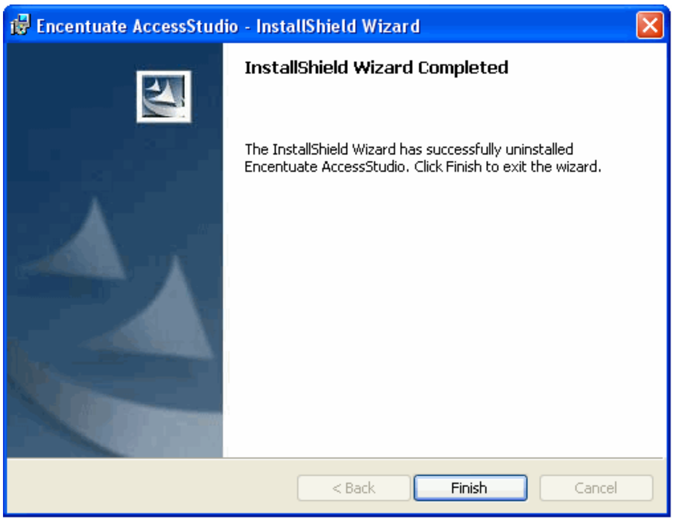
Click Remove.

❹ To confirm the uninstallation of AccessStudio, click **Remove**.The uninstallation begins when the progress window appears.


Install Encentuate AccessStudio

Uninstallation progress bar

❺ Click **Finish** after a successful uninstallation of AccessStudio.

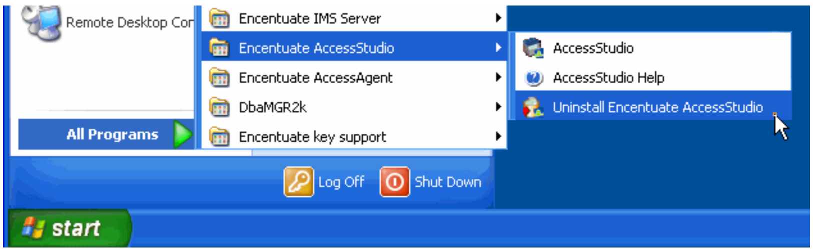

Encentuate AccessStudio - InstallShield Wizard

InstallShield Wizard Completed

The InstallShield Wizard has successfully uninstalled Encentuate AccessStudio. Click Finish to exit the wizard.

Successful uninstallation confirmation screen

Here is another way to easily uninstall AccessStudio:



Go to Start >> Encentuate AccessStudio >> Uninstall Encentuate AccessStudio.

About AccessStudio

AccessStudio is a simple yet powerful tool that enables an administrator to create and manage AccessProfiles.

Encentuate AccessAgent facilitates automatic logon, log off, and password change for a multitude of applications that require authentication. AccessAgent also automates workflows. However, no two organizations will have the same set of applications. Some may have applications that are not in the default set of applications provided by Encentuate IAM, and must be configured to be included in the set.

Each application is represented by an AccessProfile, which is a set of instructions that define the automatic logon mechanism for that particular application. To configure support for additional applications, it is necessary to create AccessProfiles for these applications.

This guide provides detailed information on AccessStudio, and a step-by-step guide to creating AccessProfiles.

Refer to the following main topics:

- Features and benefits
- How AccessStudio works
- Basic concepts
- Advanced concepts
- The AccessStudio interface

Features and benefits

AccessStudio provides you with maximum control over configuring AccessProfiles and their associated data (which includes application objects, authentication services, authentication service groups, and authentication service group links).

You can set up AccessProfiles for the following types of applications:

- Windows applications
- Web applications
- Applications that use Java applets
- Terminal applications
- Mainframe applications
- Applications with owner-drawn window screens

AccessStudio offers a set of value-added features designed to simplify your tasks, such as:

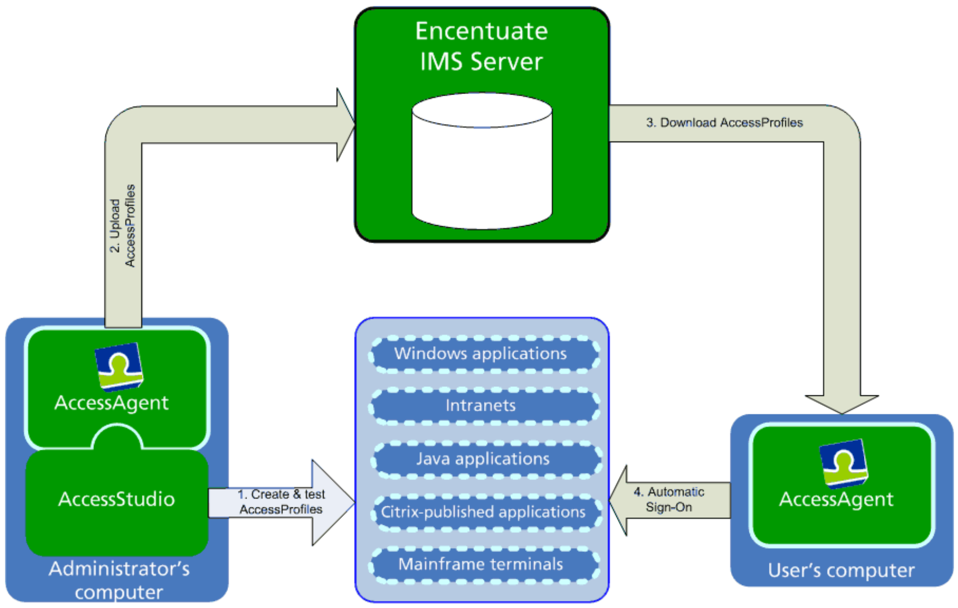
- Creation of standard and advanced modes of AccessProfiles to support requirements of varying complexity.
- User-friendly interface with multiple editors (GUI-based and XML editors) to suit your preferences.
- Flexibility in editing AccessProfiles stored in any location, including those existing in the IMS Server.
- Import existing AccessProfiles from your local installation of AccessAgent or from the IMS Server.
- Automatic validation of user-configured AccessProfile Data, thus minimizing errors.
- Ability to test and debug AccessProfiles.

How AccessStudio works

You can create AccessProfile data and save it to a file using AccessStudio. You can also download and modify AccessProfiles and their associated data from either the Encentuate IMS Server or the local installation of AccessAgent.

After creating or modifying an AccessProfile and its associated data, use the **Upload to IMS** option to publish the data to the IMS Server. After the IMS Server receives the update, the data is downloaded by the AccessAgent associated with the IMS Server. Any changes or newly-created AccessProfiles are applied to the applications in the users's system.

The following figure illustrates this process:



AccessStudio architecture

Basic concepts

Before you begin using AccessStudio, you must familiarize yourself with the concepts inherent to the creation of AccessProfiles.

AccessProfile

An AccessProfile contains instructions on handling automation for an application. An application can be an executable file (.EXE) or a Web page. An AccessProfile includes the following:

- Information to identify the application.
- Instructions for performing automatic operations, such as automatic logon or logoff for the application.
- A reference to the entity that validates the logon information for the application, known as the Authentication service.
- A reference to the entity that represents the AccessProfile's associated group, otherwise known as the Application Object.

There are two (2) kinds of AccessProfiles - *Standard* and *Advanced*. Use AccessStudio's AccessProfile Generator to create standard AccessProfiles through a series of wizard windows. Standard AccessProfiles will do for automating most applications.

For more complex AccessProfiles, you need to create advanced AccessProfiles for your applications. To understand concepts used in advanced AccessProfiles, see Advanced concepts.

Authentication service

Most applications need to validate logon information using a verification entity known as an Authentication Service. All created AccessProfiles must be associated with an authentication service.

Multiple AccessProfiles can be associated with a single authentication service. If a group of applications have been associated with the same authentication service, AccessStudio ensures that any changes made to the logon information within one application will be applied across all applications associated with the same authentication service.

Refer to the following example:

> The Yahoo! Messenger, Yahoo! Mail, and Yahoo! Chat are all different applications represented by different AccessProfiles. However, the same user name and password is used to access all three applications. The Yahoo! authentication service validates all logon information. Therefore, only one authentication service is created to represent all applications validated by the Yahoo! authentication service.
>
> When you log on to Yahoo! Messenger, you do not need to log on again when you access Yahoo! Chat and/or Yahoo! Mail. Your logon information for Yahoo! Messenger was already captured for all other Yahoo! applications, since they are all associated with the same authentication service.
>
> The same concept applies for any changes made to your logon information. For example, if you change your password using Yahoo! Mail, the new password is captured for all other Yahoo! applications.

Application

An application object is a logical representation in AccessStudio of a set of executable files (.EXE) or Web pages. An application object allows you to apply tighter control policies for a group of AccessProfiles.

In AccessStudio, one AccessProfile is created for a .EXE file or Web page, and each .EXE file or Web page is treated as an application.

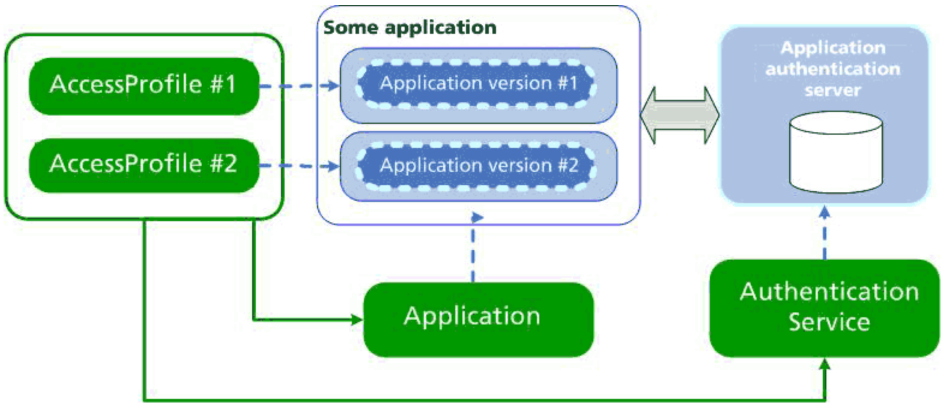
An application object handles the grouping of .EXE files and Web pages as belonging to the same entity. Each AccessProfile must be associated with an application object.

Refer to the following example:

> The Yahoo! authentication service is used by Websites mail.yahoo.com and chat.yahoo.com, as well as by Yahoo! Messenger version 5 and version 6; Each .EXE file or Web page would require its own AccessProfile. You can create up to four application objects, depending on your preferred extent of control over the automatic logon policy of the four AccessProfiles.
>
> For instance, if you want mail.yahoo.com and chat.yahoo.com to have different automatic-sign on mechanisms from the two Yahoo! Messengers, you can group them under two different application objects. Alternatively, if you require the same automatic-sign on mechanism for all four, you can group them all under a single application object.

An AccessProfile can be associated with only one application object, while a single application object can be associated with multiple AccessProfiles.

The following diagram illustrates the relationship between all entities associated with an AccessProfile.



How other entities are associated with AccessProfile

In the diagram, AccessProfiles #1 and #2 represent two different versions of the same application. These applications communicate with an application authentication server. The AccessProfiles representing each of these applications in turn, are associated with an application object and an authentication service. The authentication service has a reference to the application's actual authentication service.

Advanced concepts

To work with advanced AccessProfiles, you must familiarize yourself with the following concepts.

Skip this section if you prefer to use AccessProfile Generator to create AccessProfiles.

Standard AccessProfile

Standard AccessProfiles, also known as *Simple SSO Support,* contain all logon, password, and logoff information within single or multiple screens. Examples are the logon screens for applications, such as MSN Messenger, Yahoo! Mail, and Hotmail. Standard AccessProfiles also support most applications in different deployment scenarios.

All logon and password change information for each application should be under one AccessProfile.

Advanced AccessProfile

Advanced AccessProfiles, also known as *State Engine SSO Support,* allows you to automate operations based on various conditions. Use advanced AccessProfiles for complex logon situations, such as verification of conditions before automatic logon, greater control over what triggers an action, and the sequence of these actions.

Advanced AccessProfiles are based on a State Engine, which includes states, triggers, and actions. The interaction between these three components determines how automatic operations are managed for an AccessProfile. For more information on these three components, see State, Trigger, and Action.

State

A State indicates the current condition or status of an application (for example: signed-on status or signed-off status). You can define multiple states and can associate triggers that cause a transition from one state to another. It is also possible to provide triggers which point to the same state.

Each state is identified by a user-defined unique ID. You must define a start state which is needed to execute the state's transitions.

Trigger

A trigger is an event that causes transitions between states in a state engine (e.g, the loading of a Web page or the appearance of a window on the desktop). AccessStudio contains predefined triggers that cover an exhaustive set of requirements.

Once a trigger fires, it executes a set of actions defined by the Administrator and then causes transition to the next indicated state. See Triggers for a list of pre-defined AccessStudio triggers.

Action

An action is the process performed in response to a trigger. An example of an action is the automatic filling of user name and password details as soon as a logon window appears. When a trigger fires, the actions specified for that particular trigger are executed in a pre-defined sequence.

AccessStudio contains pre-defined actions that can be used to perform a set of operations in the application.

The interaction between states, triggers, and actions can be understood through the following example.

❶ When the MSN Messenger launches, it is in Start *state*. With the appearance of the logon window, a *trigger* is fired. Following this, the necessary *action* to auto-fill the logon information occurs.

❷ The messenger comes to an after-auto-fill *state* that is defined in the engine. A *trigger* is activated when the user clicks the **Sign in** button. The *action* to capture the user name and password information occurs.

❸ The messenger now assumes the after-capture-*state*. A *trigger* is activated when the signed-in screen appears with the contacts list, and an *action* to save this user name and password information occurs.

❹ Finally, the messenger returns to the Start *state*.

See Actions for a list of pre-defined AccessStudio Actions.

Account data

Account data is the logon information required for verification against an authentication service. The account data usually refers to the user name, password, and the authentication service that stores the logon information.

AccessStudio stores the account data in a specific format known as *Account Data Templates*. Account data templates provide information about the captured data (for example: which fields are key fields, case-sensitive, and which fields must be hidden).

A set of account data template IDs is defined in AccessStudio with each ID representing a particular type of account data. For example, the most commonly used ID (**adtid_ciuser_cspwd**) can be specified for applications that have one case-insensitive user name and one case-sensitive password. See Account Data Items and Templates for details.

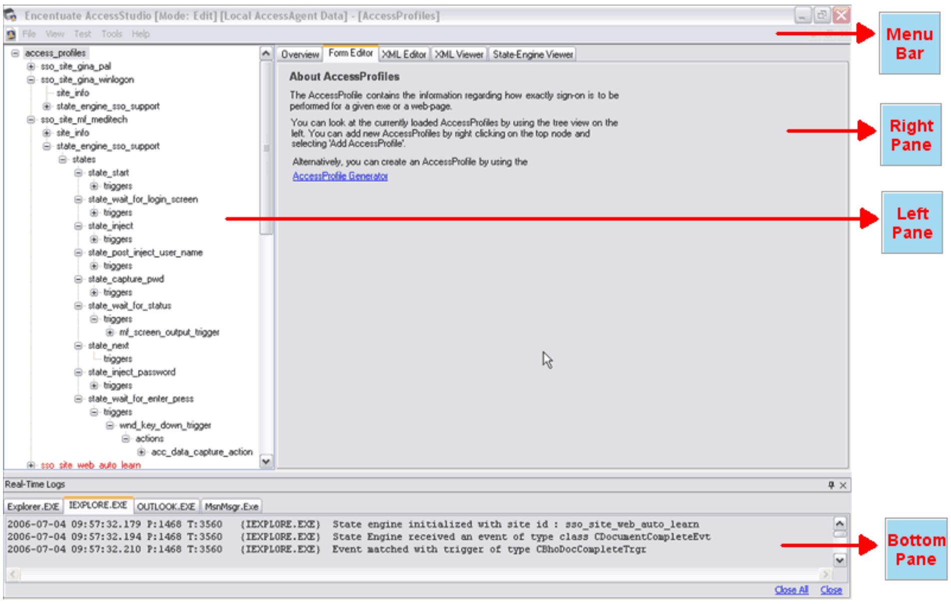
Refer to this example:

> For Yahoo! applications, the account data contains the authentication service ID (which is a user-specified name for the Yahoo! authentication service), the user name, the encrypted password, and the account data template ID. The account data template ID declares that the user name field is a key field, and that it is case-insensitive and is not a secret. Similarly for the Password field, the account data template specifies that it is not a key field, that it is case-sensitive, and that it is a secret (and therefore requires encryption).

The AccessStudio interface

AccessStudio comes with a user-friendly interface that provides complete flexibility for configuration. There are four main parts to the interface:

- Menu bar
- Left pane
- Right pane
- Bottom pane



About AccessProfiles

The AccessProfile contains the information regarding how exactly sign-on is to be performed for a given exe or a web-page.

You can look at the currently loaded AccessProfiles by using the tree view on the left. You can add new AccessProfiles by right clicking on the top node and selecting 'Add AccessProfile'.

Alternatively, you can create an AccessProfile by using the AccessProfile Generator

```
2006-07-04 09:57:32.179 P:1468 T:3560   (IEXPLORE.EXE)  State engine initialized with site id : sso_site_web_auto_learn
2006-07-04 09:57:32.194 P:1468 T:3560   (IEXPLORE.EXE)  State Engine received an event of type class CDocumentCompleteEvt
2006-07-04 09:57:32.210 P:1468 T:3560   (IEXPLORE.EXE)  Event matched with trigger of type CBhoDocCompleteTrgr
```

AccessStudio user interface parts

Menu bar

The AccessStudio menu bar contains the following menus.

Menu	Descriptions
File	The File menu contains functions for creating, opening, and saving files that store AccessProfiles. It also contains a function for importing data from the IMS Server or the AccessAgent installed on your computer.
View	The View menu contains functions for selecting an item (for example: authentication service) for configuration or viewing, or a task for execution (for example: viewing the list of account data templates).
Test	The Test menu contains functions for starting or stopping a test for detecting errors in AccessProfiles and/or their associated data.
Tools	The Tools menu contains functions for running the AccessProfile Generator and saving data related to your computer from the IMS Server to a file.
Help	The Help menu contains version information of AccessStudio.

The menu bar

Left pane

Use AccessStudio's left pane to view or modify AccessProfiles and their associated data. From the left pane, you can view an AccessProfile's authentication services, application objects, authentication service groups, authentication service group links, account data templates, and account data item templates.

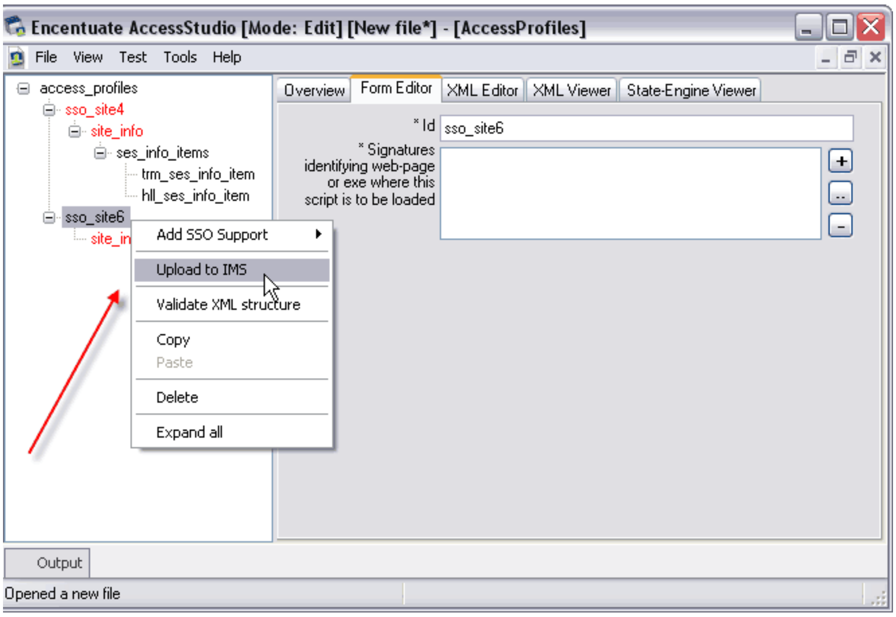
For example, when you access an AccessProfile using the **View** menu, the left pane displays the hierarchy of all data items and sub-items associated with this task. AccessProfiles are configured in the XML format.

The data items and sub-items are known as nodes and sub-nodes. When you open an AccessProfile for configuration, each added AccessProfile in the left pane represents a data-item or node. The type of AccessProfile you add under each AccessProfile represents a sub-node of that AccessProfile. The tree in the left pane typically consists of groups of nodes and sub-nodes, which can be expanded or collapsed. A plus (+) sign before a node/sub-node indicates that it is expandable.

Right-click on a node to perform either of the following actions:

- Navigate through the nodes
- Add, edit, and delete nodes
- Cut, copy, and paste nodes
- Perform related tasks and sub-tasks (for example: upload an AccessProfile to the IMS Server).

The left pane is meant to create or navigate through nodes. Use the right pane to edit existing node information.



* Id: sso_site6

Opened a new file

Navigate through the topic tree in the left pane.

Right pane

Use the right pane to view and edit information associated with a node. The right pane has five (5) tabs, three (3) of which are for viewing node information. The remaining two (2) tabs allows you to edit node information. These tabs can be grouped as follows:

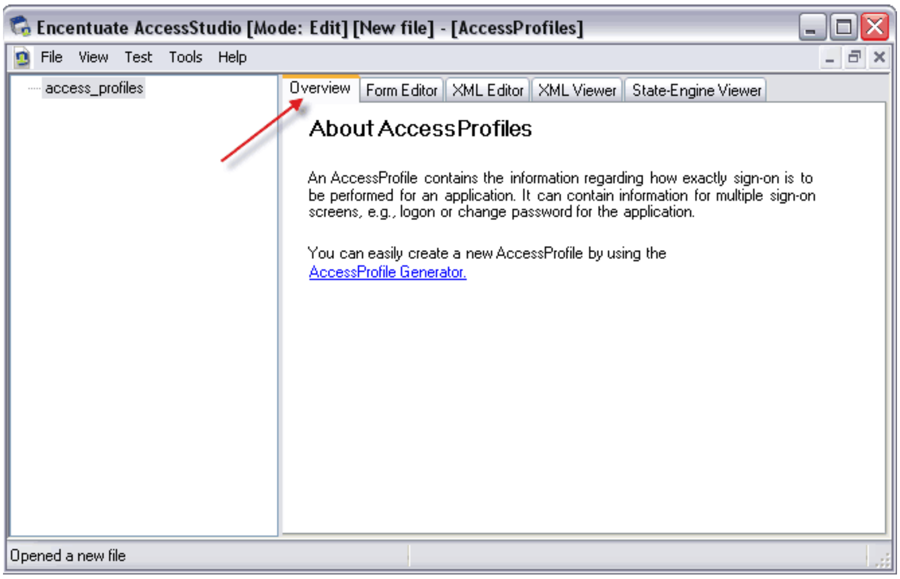
- Standard tabs

  Use the standard tabs to create and edit most AccessProfiles and their associated data.

  - Overview tab

    Use the **Overview** tab to view summary information about the node selected from the left pane. The **Overview** tab can also provide links for performing other actions, such as launching the AccessProfile Generator.

It is recommended to read the information in the **Overview** *tab before proceeding with editing the fields in the Form Editor tab.*


Encentuate AccessStudio [Mode: Edit] [New file] - [AccessProfiles]
File View Test Tools Help
access_profiles
Overview
Form Editor
XML Editor
XML Viewer
State-Engine Viewer

About AccessProfiles

An AccessProfile contains the information regarding how exactly sign-on is to be performed for an application. It can contain information for multiple sign-on screens, e.g., logon or change password for the application.

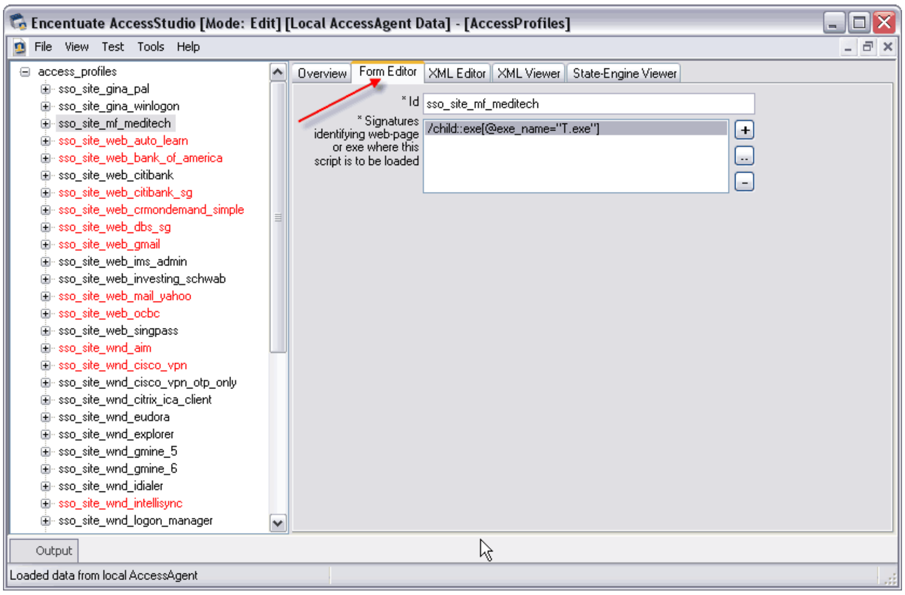
You can easily create a new AccessProfile by using the AccessProfile Generator.

Opened a new file

Overview tab

- **Form Editor tab**

  Use the Form Editor tab to add, edit and view configuration information related to the node selected from the left pane.


Encentuate AccessStudio [Mode: Edit] [Local AccessAgent Data] - [AccessProfiles]
File View Test Tools Help
Overview
Form Editor
XML Editor
XML Viewer
State-Engine Viewer
Output

access_profiles
- sso_site_gina_pal
- sso_site_gina_winlogon
- sso_site_mf_meditech
- sso_site_web_auto_learn
- sso_site_web_bank_of_america
- sso_site_web_citibank
- sso_site_web_citibank_sg
- sso_site_web_crmondemand_simple
- sso_site_web_dbs_sg
- sso_site_web_gmail
- sso_site_web_ims_admin
- sso_site_web_investing_schwab
- sso_site_web_mail_yahoo
- sso_site_web_ocbc
- sso_site_web_singpass
- sso_site_wnd_aim
- sso_site_wnd_cisco_vpn
- sso_site_wnd_cisco_vpn_otp_only
- sso_site_wnd_citrix_ica_client
- sso_site_wnd_eudora
- sso_site_wnd_explorer
- sso_site_wnd_gmine_5
- sso_site_wnd_gmine_6
- sso_site_wnd_idialer
- sso_site_wnd_intellisync
- sso_site_wnd_logon_manager

* Id	sso_site_mf_meditech
* Signatures identifying web-page or exe where this script is to be loaded	/child::exe[@exe_name="T.exe"]

Loaded data from local AccessAgent

Form Editor tab

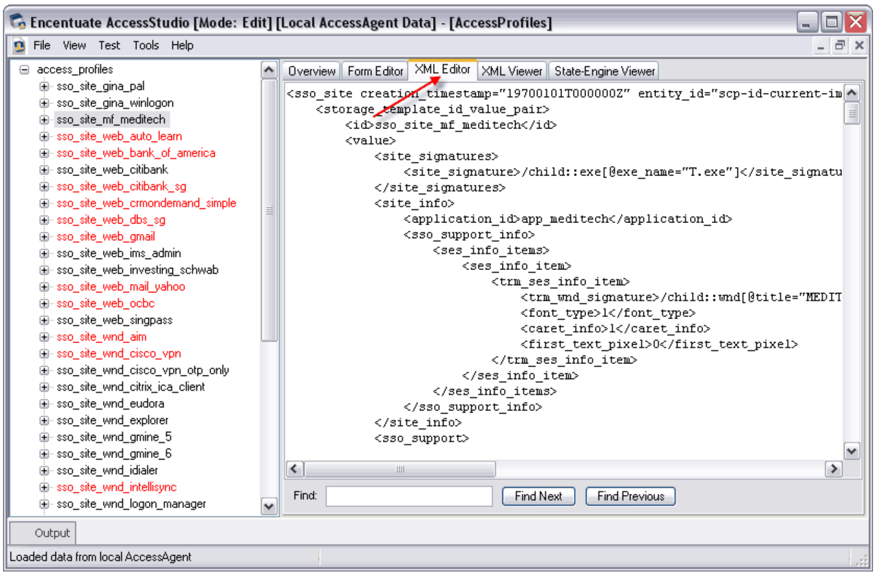
- **Advanced tabs**

  The advanced tabs are designed for users who are familiar with XML configuration.

Use these tabs only if you are familiar with XML.

- XML Editor tab

  Use the XML Editor tab to modify the actual XML code of the AccessProfile or its associated data. To validate the XML code, right-click on the node in the left pane and select **Validate XML Structure**. Nodes with erroneous XML structures will be highlighted in red on the left pane.


Encentuate AccessStudio [Mode: Edit] [Local AccessAgent Data] - [AccessProfiles]
File View Test Tools Help
Overview
Form Editor
XML Editor
XML Viewer
State-Engine Viewer
Find:
Find Next
Find Previous
Output

access_profiles
- sso_site_gina_pal
- sso_site_gina_winlogon
- sso_site_mf_meditech
- sso_site_web_auto_learn
- sso_site_web_bank_of_america
- sso_site_web_citibank
- sso_site_web_citibank_sg
- sso_site_web_crmondemand_simple
- sso_site_web_dbs_sg
- sso_site_web_gmail
- sso_site_web_ims_admin
- sso_site_web_investing_schwab
- sso_site_web_mail_yahoo
- sso_site_web_ocbc
- sso_site_web_singpass
- sso_site_wnd_aim
- sso_site_wnd_cisco_vpn
- sso_site_wnd_cisco_vpn_otp_only
- sso_site_wnd_citrix_ica_client
- sso_site_wnd_eudora
- sso_site_wnd_explorer
- sso_site_wnd_gmine_5
- sso_site_wnd_gmine_6
- sso_site_wnd_idialer
- sso_site_wnd_intellisync
- sso_site_wnd_logon_manager

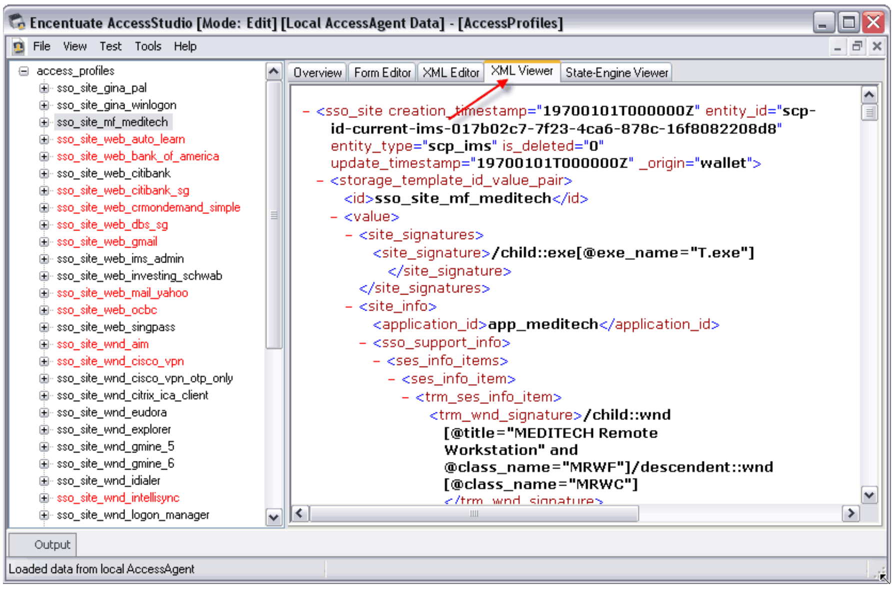
```
<sso_site creation_timestamp="19700101T000000Z" entity_id="scp-id-current-im
    <storage_template_id_value_pair>
        <id>sso_site_mf_meditech</id>
        <value>
            <site_signatures>
                <site_signature>/child::exe[@exe_name="T.exe"]</site_signatu
            </site_signatures>
            <site_info>
                <application_id>app_meditech</application_id>
                <sso_support_info>
                    <ses_info_items>
                        <ses_info_item>
                            <trm_ses_info_item>
                                <trm_wnd_signature>/child::wnd[@title="MEDIT
                                <font_type>1</font_type>
                                <caret_info>1</caret_info>
                                <first_text_pixel>0</first_text_pixel>
                            </trm_ses_info_item>
                        </ses_info_item>
                    </ses_info_items>
                </sso_support_info>
            </site_info>
            <sso_support>
```

Loaded data from local AccessAgent

XML Editor tab

- XML Viewer tab

  Use the XML Viewer tab to view a node’s actual XML code.


Encentuate AccessStudio [Mode: Edit] [Local AccessAgent Data] - [AccessProfiles]
File View Test Tools Help
Overview
Form Editor
XML Editor
XML Viewer
State-Engine Viewer
Output

access_profiles
sso_site_gina_pal
sso_site_gina_winlogon
sso_site_mf_meditech
sso_site_web_auto_learn
sso_site_web_bank_of_america
sso_site_web_citibank
sso_site_web_citibank_sg
sso_site_web_crmondemand_simple
sso_site_web_dbs_sg
sso_site_web_gmail
sso_site_web_ims_admin
sso_site_web_investing_schwab
sso_site_web_mail_yahoo
sso_site_web_ocbc
sso_site_web_singpass
sso_site_wnd_aim
sso_site_wnd_cisco_vpn
sso_site_wnd_cisco_vpn_otp_only
sso_site_wnd_citrix_ica_client
sso_site_wnd_eudora
sso_site_wnd_explorer
sso_site_wnd_gmine_5
sso_site_wnd_gmine_6
sso_site_wnd_idialer
sso_site_wnd_intellisync
sso_site_wnd_logon_manager

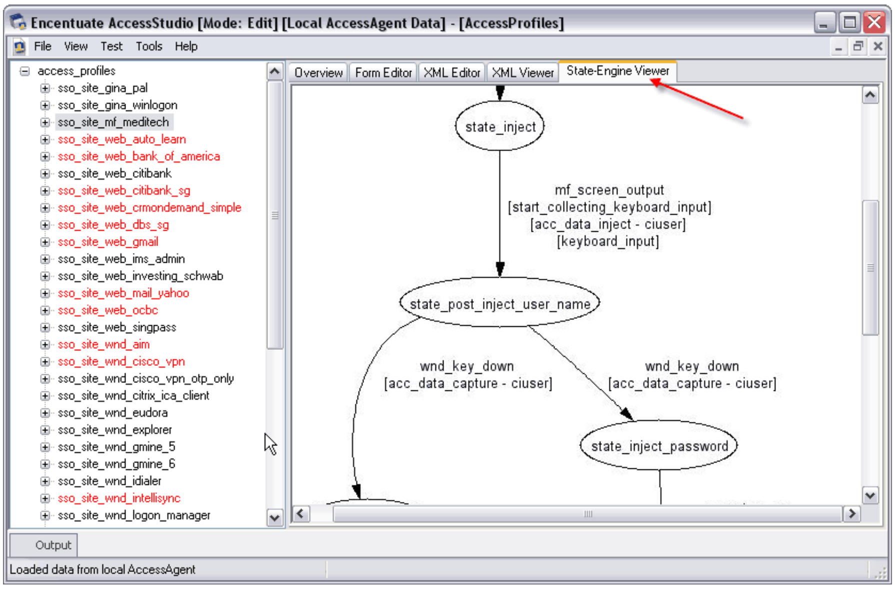
```
- <sso_site creation_timestamp="19700101T000000Z" entity_id="scp-
  id-current-ims-017b02c7-7f23-4ca6-878c-16f8082208d8"
  entity_type="scp_ims" is_deleted="0"
  update_timestamp="19700101T000000Z" _origin="wallet">
 - <storage_template_id_value_pair>
     <id>sso_site_mf_meditech</id>
   - <value>
     - <site_signatures>
         <site_signature>/child::exe[@exe_name="T.exe"]
           </site_signature>
       </site_signatures>
     - <site_info>
         <application_id>app_meditech</application_id>
       - <sso_support_info>
         - <ses_info_items>
           - <ses_info_item>
             - <trm_ses_info_item>
                 <trm_wnd_signature>/child::wnd
                   [@title="MEDITECH Remote
                   Workstation" and
                   @class_name="MRWF"]/descendent::wnd
                   [@class_name="MRWC"]
                   </trm_wnd_signature>
```

Loaded data from local AccessAgent

XML Viewer tab

- **State-engine Viewer tab**

  Use the State-Engine Viewer tab for a visual representation of the sequence of the AccessProfile's states, triggers, and actions. This tab helps you trace the dependencies between the states, triggers, and actions.


Encentuate AccessStudio [Mode: Edit] [Local AccessAgent Data] - [AccessProfiles]
File View Test Tools Help
Overview
Form Editor
XML Editor
XML Viewer
State-Engine Viewer
state_inject
mf_screen_output
[start_collecting_keyboard_input]
[acc_data_inject - ciuser]
[keyboard_input]
state_post_inject_user_name
wnd_key_down
[acc_data_capture - ciuser]
wnd_key_down
[acc_data_capture - ciuser]
state_inject_password
Output

access_profiles
sso_site_gina_pal
sso_site_gina_winlogon
sso_site_mf_meditech
sso_site_web_auto_learn
sso_site_web_bank_of_america
sso_site_web_citibank
sso_site_web_citibank_sg
sso_site_web_crmondemand_simple
sso_site_web_dbs_sg
sso_site_web_gmail
sso_site_web_ims_admin
sso_site_web_investing_schwab
sso_site_web_mail_yahoo
sso_site_web_ocbc
sso_site_web_singpass
sso_site_wnd_aim
sso_site_wnd_cisco_vpn
sso_site_wnd_cisco_vpn_otp_only
sso_site_wnd_citrix_ica_client
sso_site_wnd_eudora
sso_site_wnd_explorer
sso_site_wnd_gmine_5
sso_site_wnd_gmine_6
sso_site_wnd_idialer
sso_site_wnd_intellisync
sso_site_wnd_logon_manager

Loaded data from local AccessAgent

State-Engine Viewer tab

Bottom pane

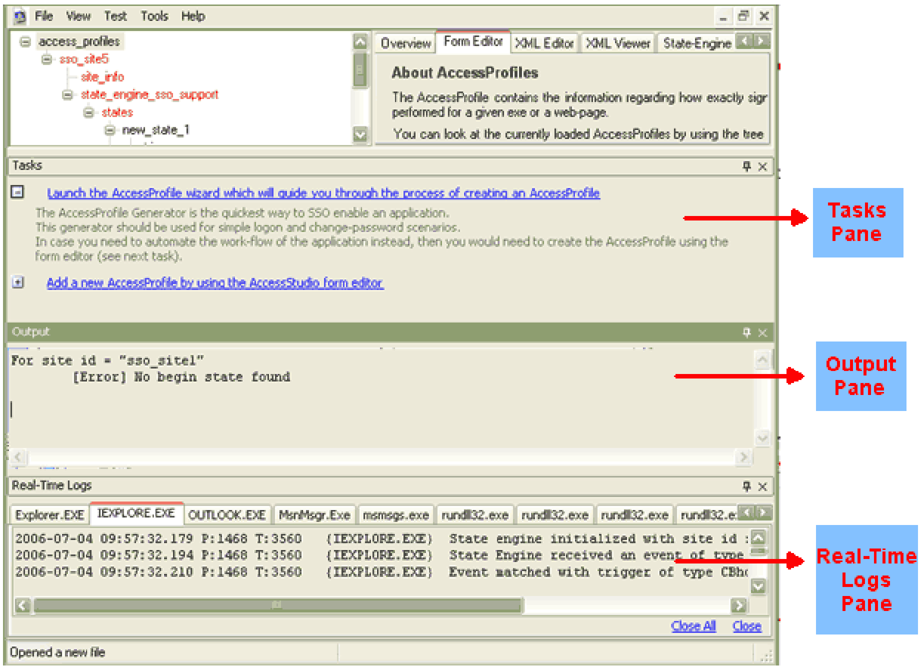
Use the bottom pane to view real-time information about the currently active task. There are three (3) types of panes: **Tasks**, **Output** and **Real-Time Logs** panes. To view any of the panes, click the *View menu >>Other windows*, and select the pane for viewing.

- Tasks pane

  Use the Tasks pane to view information on the currently performed task. Examples of information include: details on how to perform the task, what details to configure, configuration tips, technical notes, and any AccessProfile and associated data errors.

- Output pane

  Use the Output pane to view XML code validation information or Advanced AccessProfiles details.



About AccessProfiles

The AccessProfile contains the information regarding how exactly sigr
performed for a given exe or a web-page.

You can look at the currently loaded AccessProfiles by using the tree

Launch the AccessProfile wizard which will guide you through the process of creating an AccessProfile

The AccessProfile Generator is the quickest way to SSO enable an application.
This generator should be used for simple logon and change-password scenarios.
In case you need to automate the work-flow of the application instead, then you would need to create the AccessProfile using the form editor (see next task).

Add a new AccessProfile by using the AccessStudio form editor

```
For site id = "sso_site1"
        [Error] No begin state found
```

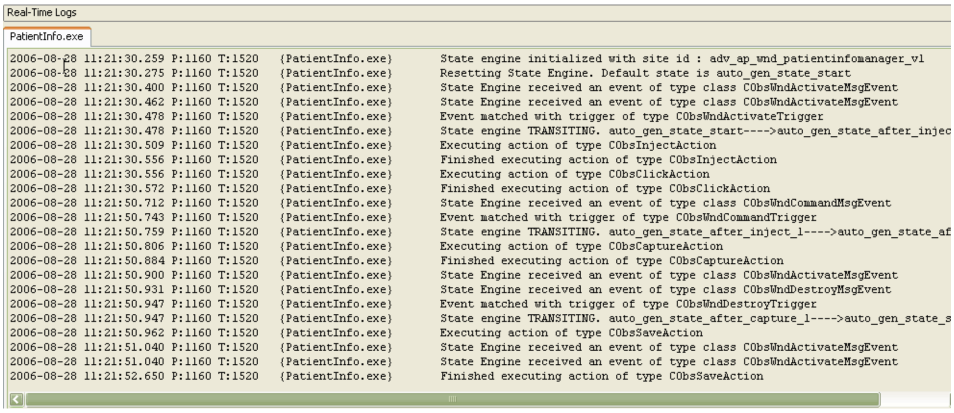
```
2006-07-04 09:57:32.179 P:1468 T:3560   (IEXPLORE.EXE)  State engine initialized with site id :
2006-07-04 09:57:32.194 P:1468 T:3560   (IEXPLORE.EXE)  State Engine received an event of type
2006-07-04 09:57:32.210 P:1468 T:3560   (IEXPLORE.EXE)  Event matched with trigger of type CBh
```

Opened a new file

Bottom pane parts

Real-Time Logs pane

Use the Real-time Logs pane to monitor the AccessProfiles running on your computer and an application's state transitions through AccessAgent. If an AccessProfile is running on your computer, a tab is created with the name of the application.



```
2006-08-28 11:21:30.259 P:1160 T:1520   {PatientInfo.exe}       State engine initialized with site id : adv_ap_wnd_patientinfomanager_v1
2006-08-28 11:21:30.275 P:1160 T:1520   {PatientInfo.exe}       Resetting State Engine. Default state is auto_gen_state_start
2006-08-28 11:21:30.400 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type class CObsWndActivateMsgEvent
2006-08-28 11:21:30.462 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type class CObsWndActivateMsgEvent
2006-08-28 11:21:30.478 P:1160 T:1520   {PatientInfo.exe}       Event matched with trigger of type CObsWndActivateTrigger
2006-08-28 11:21:30.478 P:1160 T:1520   {PatientInfo.exe}       State engine TRANSITING. auto_gen_state_start---->auto_gen_state_after_injec
2006-08-28 11:21:30.509 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsInjectAction
2006-08-28 11:21:30.556 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsInjectAction
2006-08-28 11:21:30.556 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsClickAction
2006-08-28 11:21:30.572 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsClickAction
2006-08-28 11:21:50.712 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type class CObsWndCommandMsgEvent
2006-08-28 11:21:50.743 P:1160 T:1520   {PatientInfo.exe}       Event matched with trigger of type CObsWndCommandTrigger
2006-08-28 11:21:50.759 P:1160 T:1520   {PatientInfo.exe}       State engine TRANSITING. auto_gen_state_after_inject_1---->auto_gen_state_af
2006-08-28 11:21:50.806 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsCaptureAction
2006-08-28 11:21:50.884 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsCaptureAction
2006-08-28 11:21:50.900 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type class CObsWndActivateMsgEvent
2006-08-28 11:21:50.931 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type class CObsWndDestroyMsgEvent
2006-08-28 11:21:50.947 P:1160 T:1520   {PatientInfo.exe}       Event matched with trigger of type CObsWndDestroyTrigger
2006-08-28 11:21:50.947 P:1160 T:1520   {PatientInfo.exe}       State engine TRANSITING. auto_gen_state_after_capture_1---->auto_gen_state_s
2006-08-28 11:21:50.962 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsSaveAction
2006-08-28 11:21:51.040 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type class CObsWndActivateMsgEvent
2006-08-28 11:21:51.040 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type class CObsWndActivateMsgEvent
2006-08-28 11:21:52.650 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsSaveAction
```

Real-Time Logs screen

Standard AccessProfiles

Use AccessStudio to create new, import and view existing AccessProfiles from your IMS Server or the AccessAgent installed on your computer. This chapter contains procedures for maintaining standard AccessProfiles in AccessStudio.

Before you begin using AccessStudio, it is recommended that you review the concepts provided in About AccessStudio of this guide.

Standard AccessProfiles, also known as *Simple SSO Support,* contain all logon, password, and logoff information within single or multiple screens. Examples are the logon screens for applications, such as MSN Messenger, Yahoo! Mail, and Hotmail. Standard AccessProfiles also support most applications in different deployment scenarios.

You can create standard AccessProfiles using the AccessProfile Generator.

This chapter covers the following topics:

- Creating standard AccessProfiles (AccessProfile Generator)
- Editing standard AccessProfile

Creating standard AccessProfiles (AccessProfile Generator)

Use the AccessProfile Generator to create an AccessProfile using a step-by-step wizard.

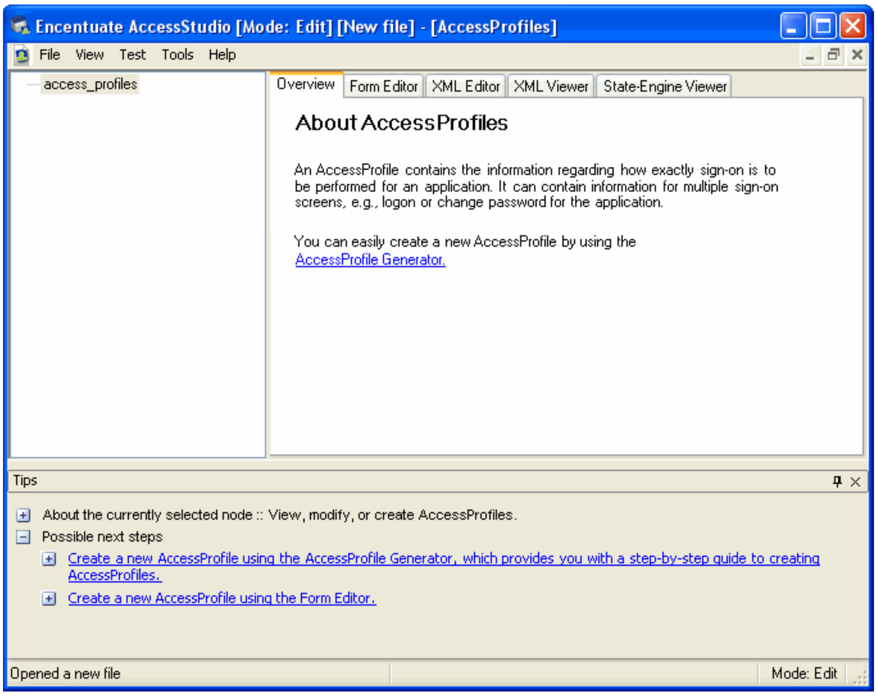
When you create an AccessProfile using the AccessProfile Generator, the wizard automatically creates the application object and the authentication service for the AccessProfile. See Basic concepts for more information on application objects and authentication services.

Using the AccessProfile Generator for Windows applications

Windows applications are (Win32, 16 bit) applications that run on Windows platform, like Outlook or Lotus Notes.

To create AccessProfiles using the AccessProfile Generator for Windows applications:

❶ Open AccessStudio. Go to *Start >> All Programs >> Encentuate AccessStudio >> AccessStudio*.



About AccessProfiles

An AccessProfile contains the information regarding how exactly sign-on is to be performed for an application. It can contain information for multiple sign-on screens, e.g., logon or change password for the application.

You can easily create a new AccessProfile by using the AccessProfile Generator.

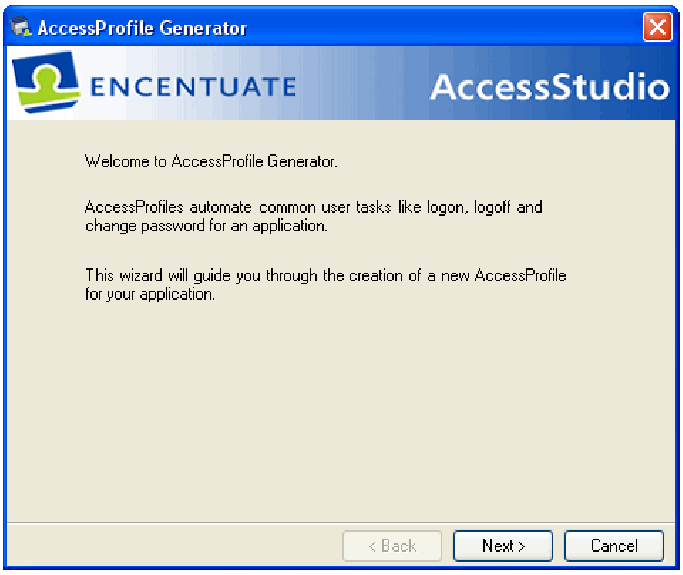
About the currently selected node :: View, modify, or create AccessProfiles.
Possible next steps
Create a new AccessProfile using the AccessProfile Generator, which provides you with a step-by-step guide to creating AccessProfiles.
Create a new AccessProfile using the Form Editor.

Opened a new file

Overview tab

❷ Start the AccessProfile Generator using any of the following instructions:

- Right-click the access_profiles node in the *Left Pane >> Start AccessProfile Generator...*
- Click the AccessProfile Generator link displayed on the **Overview** tab on the right pane
- *Go to Tools menu >> Start AccessProfile Generator.*



Welcome to AccessProfile Generator.

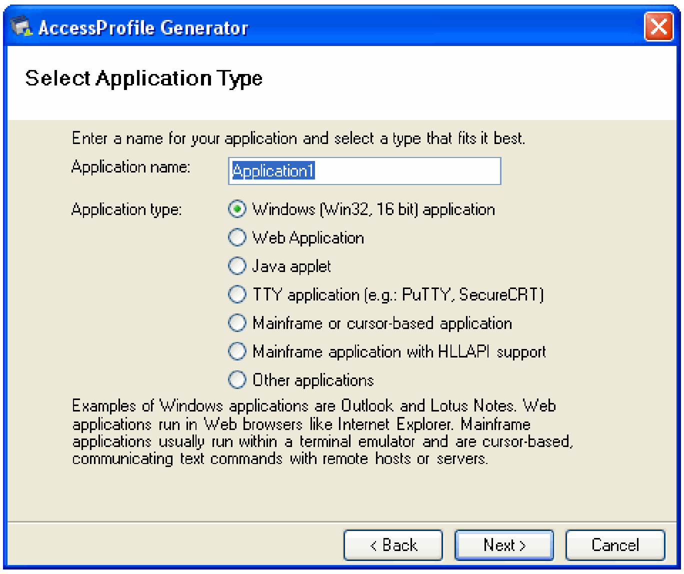
AccessProfiles automate common user tasks like logon, logoff and change password for an application.

This wizard will guide you through the creation of a new AccessProfile for your application.

Welcome screen

❸ Click **Next >** when you see the Welcome screen.

❹ Open the application logon screen that will require an AccessProfile. Once you have opened the application screen or Web page, click **Next >** to proceed.



Select Application Type

Enter a name for your application and select a type that fits it best.

Application name: Application1

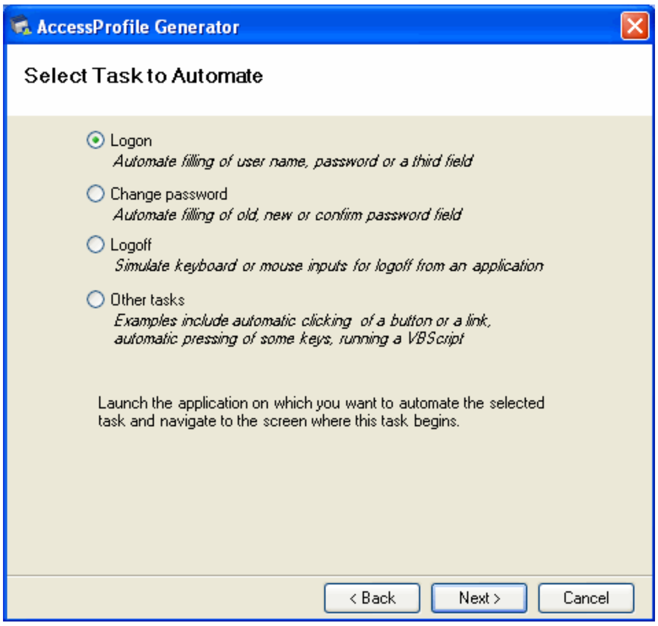
Application type:
- [✓ Windows (Win32, 16 bit) application]
- Web Application
- Java applet
- TTY application (e.g.: PuTTY, SecureCRT)
- Mainframe or cursor-based application
- Mainframe application with HLLAPI support
- Other applications

Examples of Windows applications are Outlook and Lotus Notes. Web applications run in Web browsers like Internet Explorer. Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

Select Windows (win332, 16bit) application

❺ Enter a unique name for the application and select **Windows application** as the application type. Click **Next >** to proceed.

❻ Specify the task you want to automate by marking the appropriate radio button. Click **Next >** to proceed.


AccessProfile Generator

Select Task to Automate

(● Logon) *Automate filling of user name, password or a third field*

○ Change password *Automate filling of old, new or confirm password field*

○ Logoff *Simulate keyboard or mouse inputs for logoff from an application*

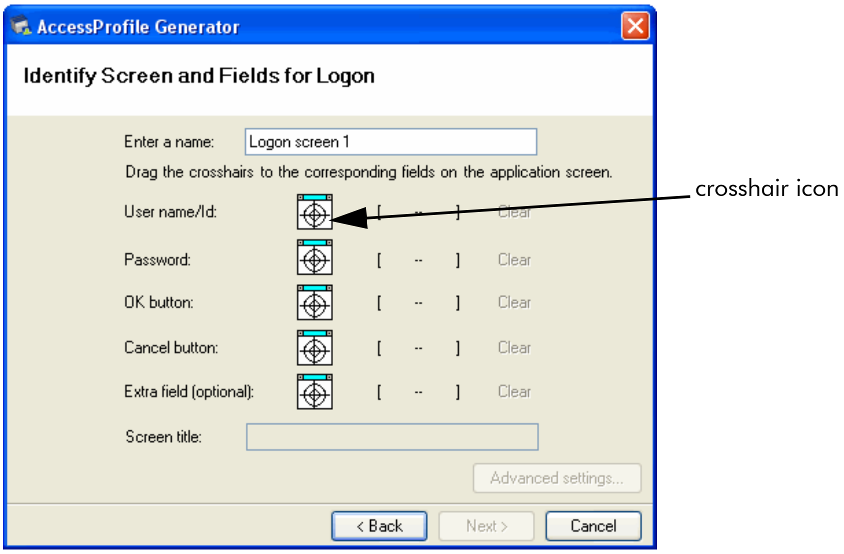
○ Other tasks *Examples include automatic clicking of a button or a link, automatic pressing of some keys, running a VBScript*

Launch the application on which you want to automate the selected task and navigate to the screen where this task begins.

< Back | Next > | Cancel

Select task to automate, then click Next >

❼ Enter a unique name for the screen or Web page to capture.


AccessProfile Generator

Identify Screen and Fields for Logon

Enter a name: Logon screen 1

Drag the crosshairs to the corresponding fields on the application screen.

User name/Id:	[--]	Clear
Password:	[--]	Clear
OK button:	[--]	Clear
Cancel button:	[--]	Clear
Extra field (optional):	[--]	Clear

Screen title:

Advanced settings...

< Back | Next > | Cancel

The Identify Screen and Fields for Logon dialog box

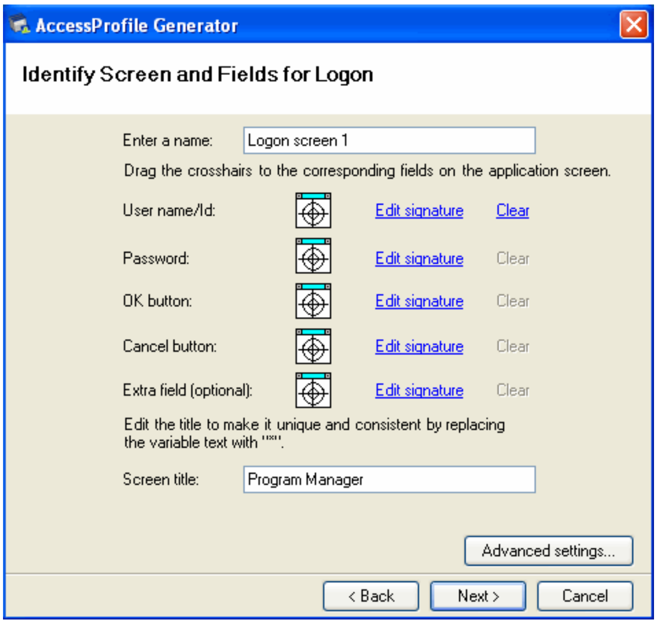
❽ Based on your selected task, capture identification information for the fields on the application screen.

The fields available for each task to be automated will vary.

❾ Click on a crosshair icon from the AccessProfile Generator, then drag and drop the icon on the matching field in the application screen or Web page. As you drag the crosshair icon to the application, AccessProfile Generator marks the field or button that can be captured. Once the crosshair icon is positioned over the field, release the mouse button.

Captured user credentials are translated into XPaths. XPath (XML Path Language) is a language that facilitates XML document navigation to select elements and attributes. These XPaths are then communicated into AccessAgent. The next time the same fields are presented, AccessAgent automatically supplies the user credentials in their respective fields. See AccessStudio Advanced for details on XPath.

If the field was captured successfully, the **Clear** option is activated. The screen title field retrieves its default screen name from the application. Click **Clear** to undo the capture.


AccessProfile Generator

Identify Screen and Fields for Logon

Enter a name: Logon screen 1

Drag the crosshairs to the corresponding fields on the application screen.

User name/Id:	Edit signature	Clear
Password:	Edit signature	Clear
OK button:	Edit signature	Clear
Cancel button:	Edit signature	Clear
Extra field (optional):	Edit signature	Clear

Edit the title to make it unique and consistent by replacing the variable text with "*".

Screen title: Program Manager

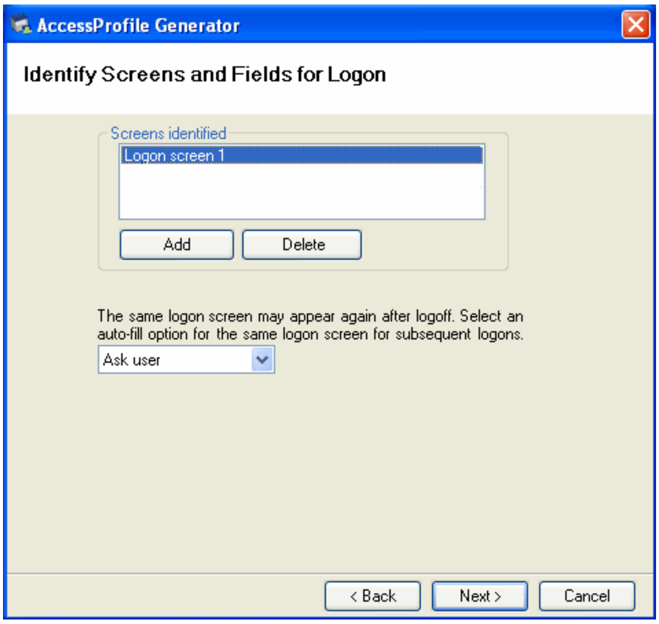
Click and drag the crosshair to the application data to capture.

Click **Edit signature** *to modify the control signature or XPath of the field you just captured. See AccessStudio Advanced for details on XPath.*

❿ Use the **Extra Field** crosshair icon to capture an additional field unique to the application, if available for the selected task.

The **Extra Field** *can be a dropdown menu, or any domain field group.*

⓫ After capturing all the application fields, click **Next >** to proceed.



AccessProfile Generator
Identify Screens and Fields for Logon
Screens identified
Logon screen 1
Add
Delete
The same logon screen may appear again after logoff. Select an auto-fill option for the same logon screen for subsequent logons.
Ask user
< Back
Next >
Cancel

Select whether you still want to capture another screen for logon or not.

To remove the previously captured screen, select the screen title in the list box and click **Delete**.

For logon screens, select the default settings for similar application screens. The options are: **Ask User**, **Do not auto-fill**, **Auto-fill**, and **Auto-fill and submit**. Based on the selection, AccessStudio will automate or not automate similar logon screens as set in this AccessProfile.

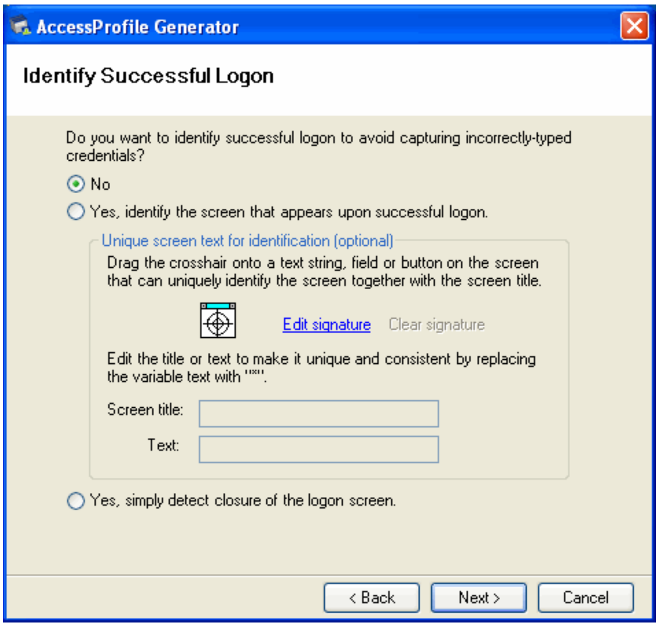
⓬ Click **Next >** to proceed.

⓭ Specify whether you want AccessStudio to identify the successful completion of the task (for example: logon, logoff, change password, etc.).

The options are **No** (no success screen or message appears), **Yes, identify the screen that appears upon successful logon**.

If you selected **Yes, identify the screen that appears upon successful logon**, drag and drop the crosshair icon on the success application screen or Web page. Once the crosshair icon is positioned over the screen or Web page, release the mouse button. Based on the captured item, you can also enter the screen title or text.

Click **Next >** to proceed.



AccessProfile Generator

Identify Successful Logon

Do you want to identify successful logon to avoid capturing incorrectly-typed credentials?

- [✓ No]
- Yes, identify the screen that appears upon successful logon.

Unique screen text for identification (optional)

Drag the crosshair onto a text string, field or button on the screen that can uniquely identify the screen together with the screen title.

Edit signature Clear signature

Edit the title or text to make it unique and consistent by replacing the variable text with "*".

Screen title:

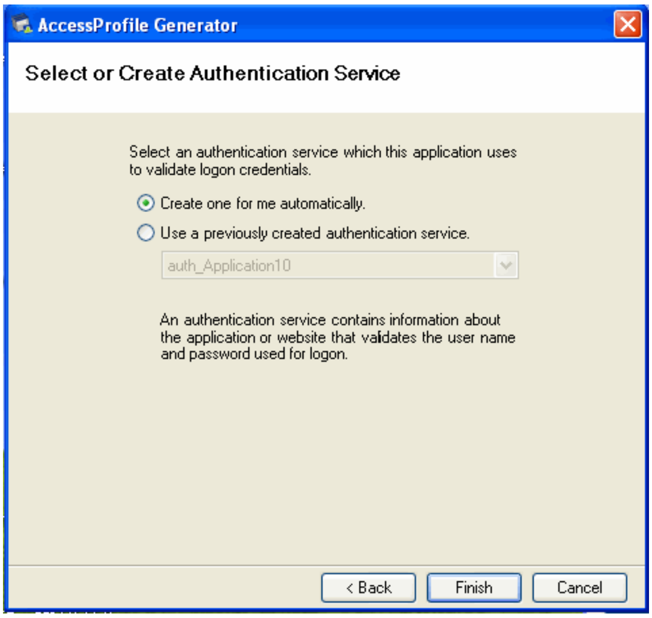
Text:

- Yes, simply detect closure of the logon screen.

< Back Next > Cancel

Specify if you want successful logon credentials or not.

⓮ Select or create authentication service.



AccessProfile Generator

Select or Create Authentication Service

Select an authentication service which this application uses to validate logon credentials.

- [✓ Create one for me automatically.]
- Use a previously created authentication service.

auth_Application10

An authentication service contains information about the application or website that validates the user name and password used for logon.

< Back Finish Cancel

Select or Create Authentication Service dialog box.

⓯ Click **Finish** to return to the AccessStudio user interface.

Test all the AccessProfiles before uploading to IMS.

⓰ Upload the AccessProfile to IMS Server to activate it. In the left pane, right-click on the AccessProfile, and select **Upload to IMS**. Click **Yes** when the **IMS Upload Confirmation** appears.

Another message box appears, indicating the success or failure of the upload.

Using the AccessProfile Generator for Web applications

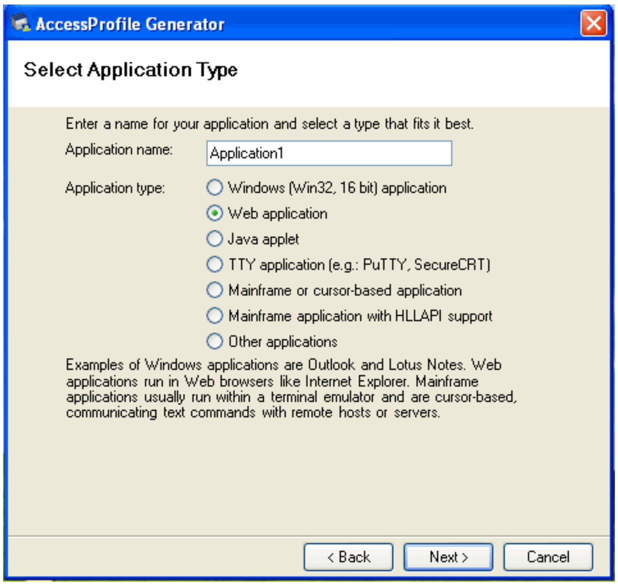
Web applications run in Web browsers like Internet Explorer.

To create AccessProfiles using the AccessProfile Generator for Web applications:

❶ See Using the AccessProfile Generator for Windows applications for the first three steps on starting the AccessProfile Generator.

❷ Once you've started the AccessProfile Generator, open the application logon screen that will require an AccessProfile.

❸ Once you have opened the application screen or Web page, click **Next >** to proceed.



Enter a name for your application and select a type that fits it best.

Application name: Application1

Application type:
- Windows (Win32, 16 bit) application
- [✓ Web application]
- Java applet
- TTY application (e.g.: PuTTY, SecureCRT)
- Mainframe or cursor-based application
- Mainframe application with HLLAPI support
- Other applications

Examples of Windows applications are Outlook and Lotus Notes. Web applications run in Web browsers like Internet Explorer. Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

Select Web application

4. Enter a unique **Application name** and select **Web application** as the application type. Click **Next** > to proceed.

5. Specify the task you want to automate by marking the appropriate radio button. Click **Next** > to proceed.

6. Enter a unique name for the screen or Web page to capture.

7. Based on your selected task, capture identification information for the fields on the application screen.

The fields available for each task to be automated will vary.

---

8. Click on a crosshair icon from the AccessProfile Generator, then drag and drop the icon on the matching field in the application screen or Web page. As you drag the crosshair icon to the application, AccessProfile Generator marks the field or button that can be captured. Once the crosshair icon is positioned over the field, release the mouse button.

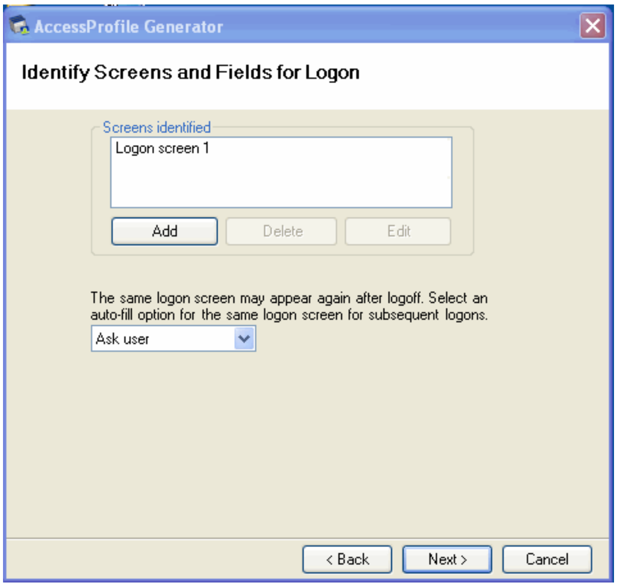
   If the field was captured successfully, the **Clear** option is activated. The screen title field retrieves its default screen name from the application. Click **Clear** to undo the capture.

9. Use the **Extra Field** crosshair icon to capture an additional field unique to the application, if available for the selected task.

10. After capturing all the application fields, click **Next** > to proceed.

11. If there are additional screens to be captured for the application, mark the **Yes** radio button, and repeat steps 7 to 9 in this procedure. Otherwise, leave the default selection at **No**.

    To remove the previously captured screen, select the screen title in the list box and click **Delete**.



Logon screen 1

The same logon screen may appear again after logoff. Select an auto-fill option for the same logon screen for subsequent logons.

Ask user

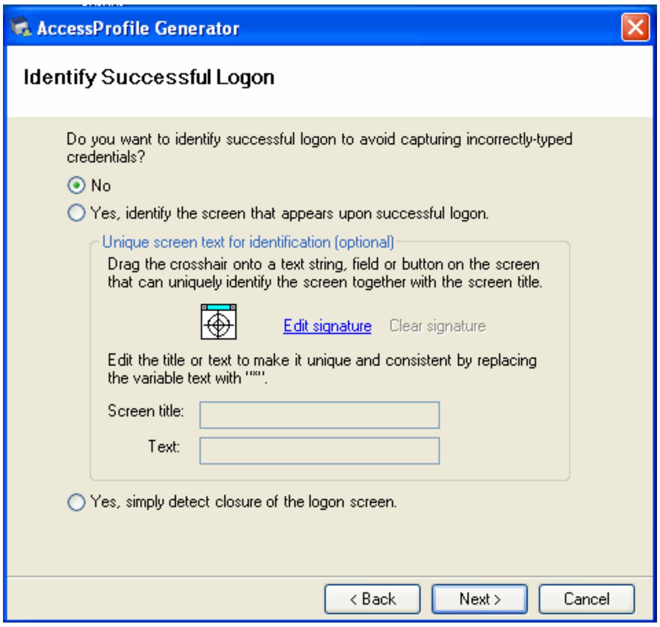
Select the screen/s identified and click Add, Delete or Edit.

For logon screens, select the default settings for similar application screens or Web pages. The options are: **Ask User**, **Do not auto-fill**, **Auto-fill**, and **Auto-fill and submit**. Based on the selection, AccessStudio will automate or not automate similar logon screens as set in this AccessProfile.

⓭ Click **Next >** to proceed.

⓮ Specify whether you want AccessStudio to identify the successful completion of the task (for example: logon, logoff, change password, etc.).

The options are **No** (no success screen or message appears), **Yes, identify the screen that appears upon successful logon**.


AccessProfile Generator

Identify Successful Logon

Do you want to identify successful logon to avoid capturing incorrectly-typed credentials?

[✓ No]

Yes, identify the screen that appears upon successful logon.

Unique screen text for identification (optional)

Drag the crosshair onto a text string, field or button on the screen that can uniquely identify the screen together with the screen title.

Edit signature Clear signature

Edit the title or text to make it unique and consistent by replacing the variable text with "*".

Screen title:

Text:

Yes, simply detect closure of the logon screen.

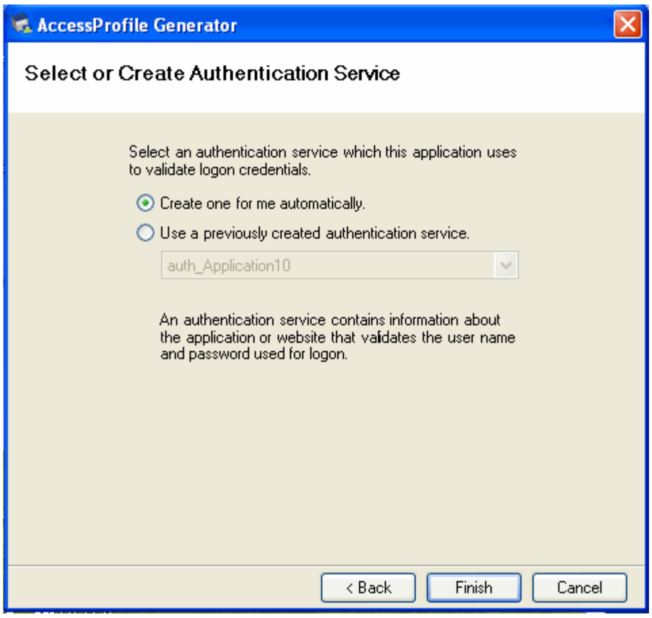
< Back Next > Cancel

Specify if you want successful logon credentials or not.

If you selected **Yes, identify the screen that appears**, drag and drop the crosshair icon on the success application screen or Web page. Once the crosshair icon is positioned over the screen or Web page, release the mouse button. Based on the captured item, you can also enter the screen title or text.

Click **Next >** to proceed.

⓯ Select or create an authentication service.


AccessProfile Generator

Select or Create Authentication Service

Select an authentication service which this application uses to validate logon credentials.

[✓ Create one for me automatically.]

Use a previously created authentication service.

auth_Application10

An authentication service contains information about the application or website that validates the user name and password used for logon.

< Back Finish Cancel

Select or Create Authentication Service dialog box.

16. Click **Finish** to return to the AccessStudio user interface.

Test all the AccessProfiles before uploading to IMS.

17. Upload the AccessProfile to IMS Server to activate it. In the left pane, right-click on the AccessProfile, and select **Upload to IMS**. Click **Yes** when the **IMS Upload Confirmation** appears.

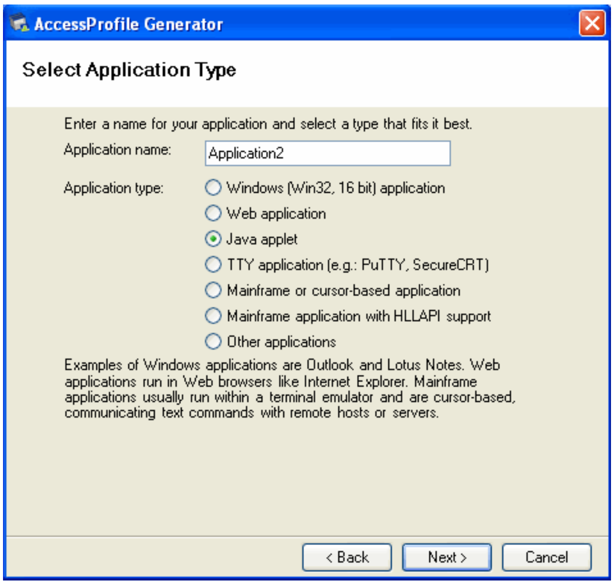
    Another message box appears, indicating the success or failure of the upload.

Using the AccessProfile Generator for applications that use Java applets

The AccessProfile Generator only works with applets that run on Sun JVM version 1.2 onwards.

To create AccessProfiles using the AccessProfile Generator for applications that use Java applets:

1. See Using the AccessProfile Generator for Windows applications for the first three steps on starting the AccessProfile Generator.
2. Open the application logon screen that will require an AccessProfile.
3. Once you have opened the application screen or Web page, click **Next >** to proceed.
4. Enter a unique name for the application and select **Java applet** as the application type. Click **Next >** to proceed.



Select Application Type

Enter a name for your application and select a type that fits it best.

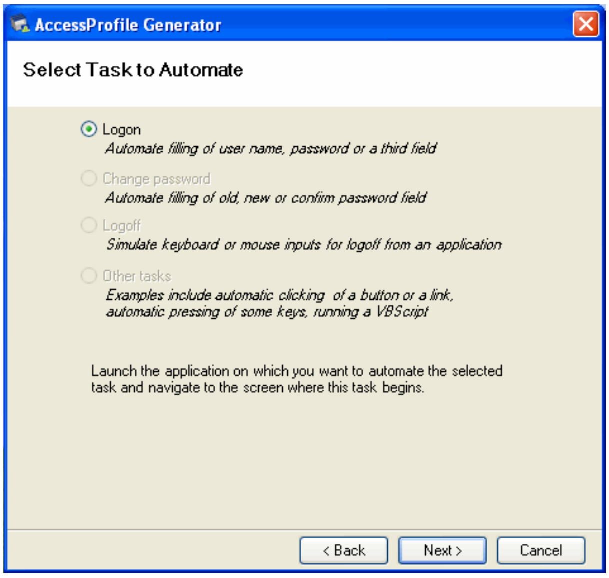
Application name: Application2

Application type:
- ( ) Windows (Win32, 16 bit) application
- ( ) Web application
- [✓ Java applet]
- ( ) TTY application (e.g.: PuTTY, SecureCRT)
- ( ) Mainframe or cursor-based application
- ( ) Mainframe application with HLLAPI support
- ( ) Other applications

Examples of Windows applications are Outlook and Lotus Notes. Web applications run in Web browsers like Internet Explorer. Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

Select Java applet as the application type

❺ Click **Next >** to proceed.



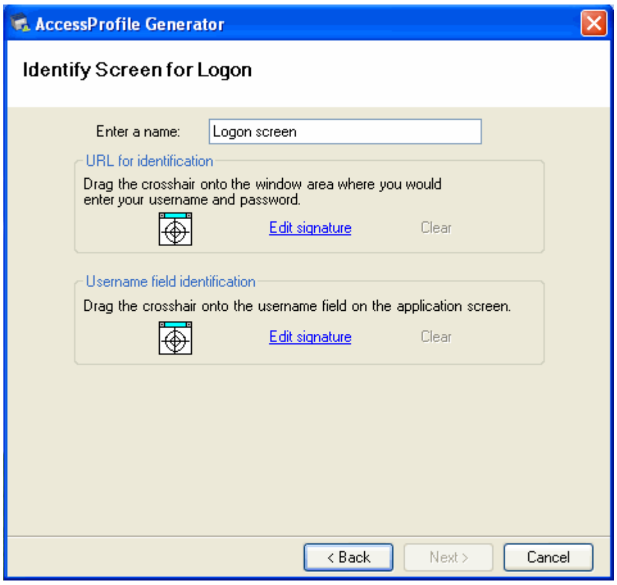
Select Task to Automate

- [✓ Logon]
  Automate filling of user name, password or a third field
- ( ) Change password
  Automate filling of old, new or confirm password field
- ( ) Logoff
  Simulate keyboard or mouse inputs for logoff from an application
- ( ) Other tasks
  Examples include automatic clicking of a button or a link, automatic pressing of some keys, running a VBScript

Launch the application on which you want to automate the selected task and navigate to the screen where this task begins.

Logon is the only active option by default

❻ Enter a unique name for the screen or Web page to capture.


AccessProfile Generator

Identify Screen for Logon

Enter a name: Logon screen

URL for identification

Drag the crosshair onto the window area where you would enter your username and password.

Username field identification

Drag the crosshair onto the username field on the application screen.

Enter a unique name for the data you need to capture.

❼ Based on your selected task, capture identification information for the fields on the application screen.

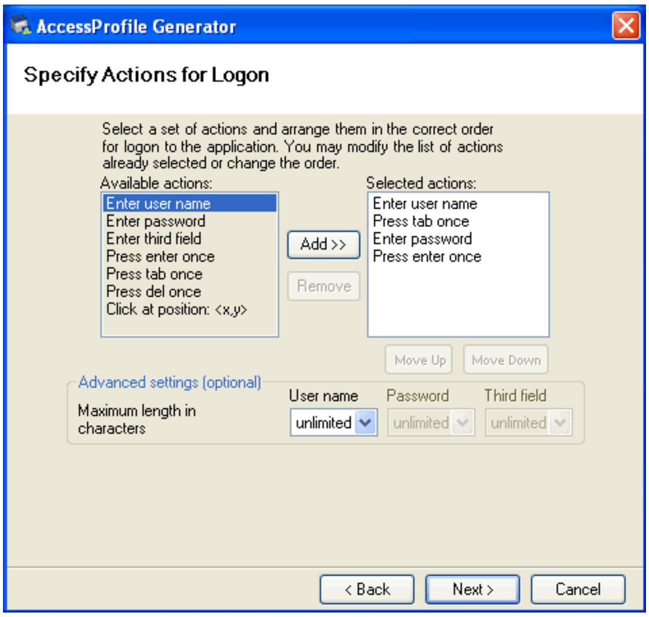
The fields available for each task to be automated will vary.

❽ Click on a crosshair icon from the AccessProfile Generator, then drag and drop the icon on the matching field in the application screen or Web page. As you drag the crosshair icon to the application, AccessProfile Generator marks the field or button that can be captured. Once the crosshair icon is positioned over the field, release the mouse button.

If the field was captured successfully, the **Clear** option is activated. The screen title field retrieves its default screen name from the application. Click **Clear** to undo the capture.

❽ After capturing all the application fields, click **Next >** to proceed.

❾ Specify Actions for Logon.


AccessProfile Generator

Specify Actions for Logon

Select a set of actions and arrange them in the correct order for logon to the application. You may modify the list of actions already selected or change the order.

Available actions:
- Enter user name
- Enter password
- Enter third field
- Press enter once
- Press tab once
- Press del once
- Click at position: <x,y>

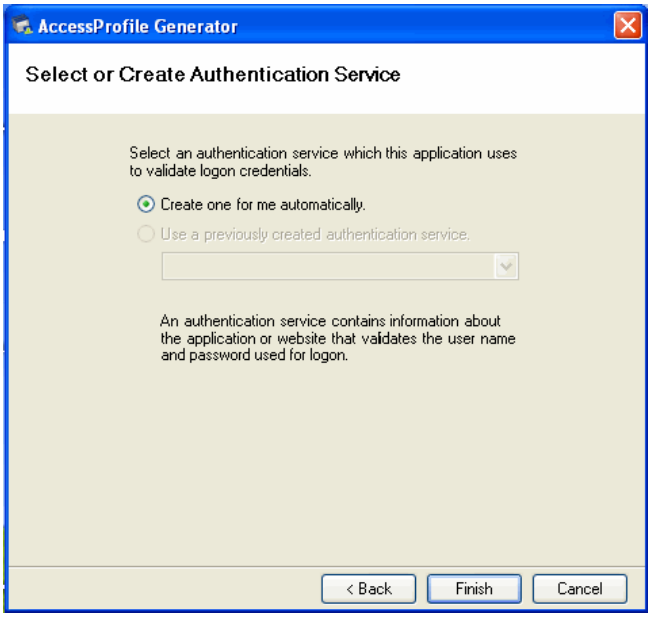
Selected actions:
- Enter user name
- Press tab once
- Enter password
- Press enter once

Advanced settings (optional)

	User name	Password	Third field
Maximum length in characters	unlimited	unlimited	unlimited

Select available actions from the left pane and click Add>>.

⓬ Click **Next >** to proceed.


AccessProfile Generator

Select or Create Authentication Service

Select an authentication service which this application uses to validate logon credentials.

[✓ Create one for me automatically.]

Use a previously created authentication service.

An authentication service contains information about the application or website that validates the user name and password used for logon.

The option Create one for me automatically is selected by default.

⓰ Click **Finish** to return to the AccessStudio user interface.

Test all the AccessProfiles before uploading to IMS.

⓱ Upload the AccessProfile to IMS Server to activate it. In the left pane, right-click on the AccessProfile, and select **Upload to IMS**. Click **Yes** when the **IMS Upload Confirmation** appears.

Another message box appears, indicating the success or failure of the upload.

Using the AccessProfile Generator for TTY applications

TTY is short for a **terminal emulator, terminal application, term**. TTY is a program that emulates a video terminal within some other display architecture. Though typically synonymous with a command line shell or text terminal, the term terminal covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window.

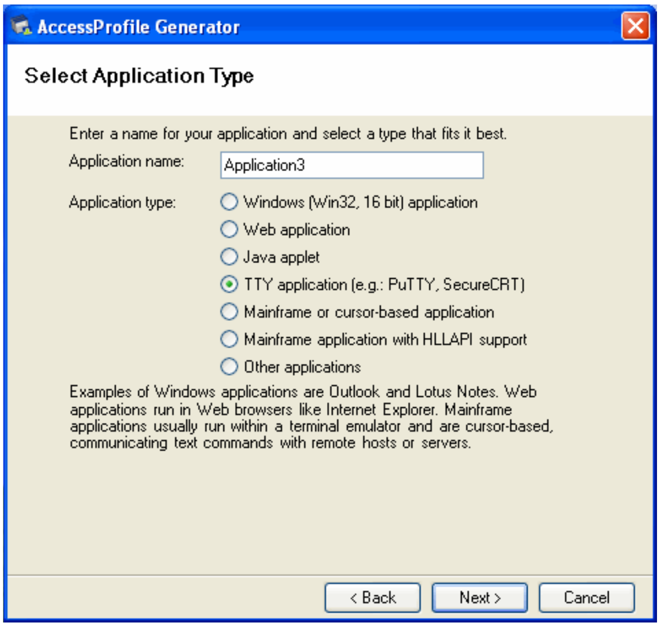
Examples of TTY applications are PuTTY and SecureCRT.

To create AccessProfiles using the AccessProfile Generator for TTY applications:

❶ See Using the AccessProfile Generator for Windows applications for the first three steps on starting the AccessProfile Generator.

❷ Open the application logon screen that will require an AccessProfile.

❸ Once you have opened the application screen or Web page, click **Next** > to proceed.


AccessProfile Generator

Select Application Type

Enter a name for your application and select a type that fits it best.

Application name: Application3

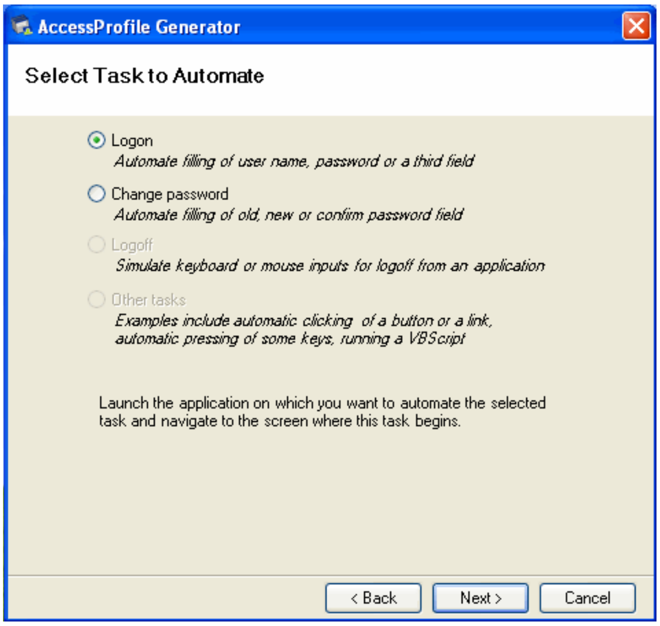
Application type:
- ( ) Windows (Win32, 16 bit) application
- ( ) Web application
- ( ) Java applet
- [✓] TTY application (e.g.: PuTTY, SecureCRT)
- ( ) Mainframe or cursor-based application
- ( ) Mainframe application with HLLAPI support
- ( ) Other applications

Examples of Windows applications are Outlook and Lotus Notes. Web applications run in Web browsers like Internet Explorer. Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

Select TTY application.

❹ Enter a unique name for the application and select **TTY application** as the application type. Click **Next >** to proceed.

❺ Specify the task you want to automate by marking the appropriate radio button. Click **Next >** to proceed.


AccessProfile Generator

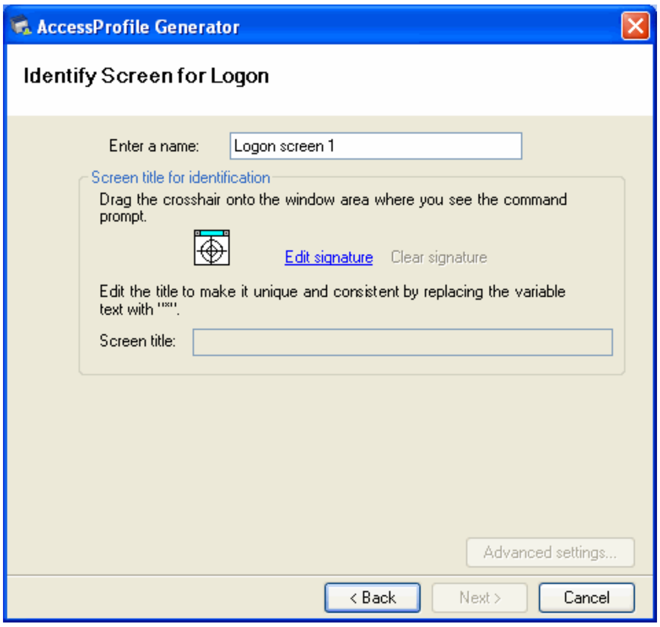
Select Task to Automate

- [✓] Logon
  Automate filling of user name, password or a third field
- ( ) Change password
  Automate filling of old, new or confirm password field
- ( ) Logoff
  Simulate keyboard or mouse inputs for logoff from an application
- ( ) Other tasks
  Examples include automatic clicking of a button or a link, automatic pressing of some keys, running a VBScript

Launch the application on which you want to automate the selected task and navigate to the screen where this task begins.

By default, only the Log on and Change password options are activated

❻ Enter a unique name for the screen or Web page to capture.


AccessProfile Generator
Identify Screen for Logon
Enter a name:
Logon screen 1
Screen title for identification
Drag the crosshair onto the window area where you see the command prompt.
Edit signature
Clear signature
Edit the title to make it unique and consistent by replacing the variable text with "*".
Screen title:
Advanced settings...
< Back
Next >
Cancel

Enter a unique name for the application data you need to capture

❼ Based on your selected task, capture identification information for the fields on the application screen.

The fields available for each task to be automated will vary.

❽ Click on a crosshair icon from the AccessProfile Generator, then drag and drop the icon on the matching field in the application screen or Web page. As you drag the crosshair icon to the application, AccessProfile Generator marks the field or button that can be captured. Once the crosshair icon is positioned over the field, release the mouse button.

If the field was captured successfully, the **Clear** option is activated. The screen title field retrieves its default screen name from the application. Click **Clear** to undo the capture.

❾ After capturing all the application fields, click **Next** > to proceed.

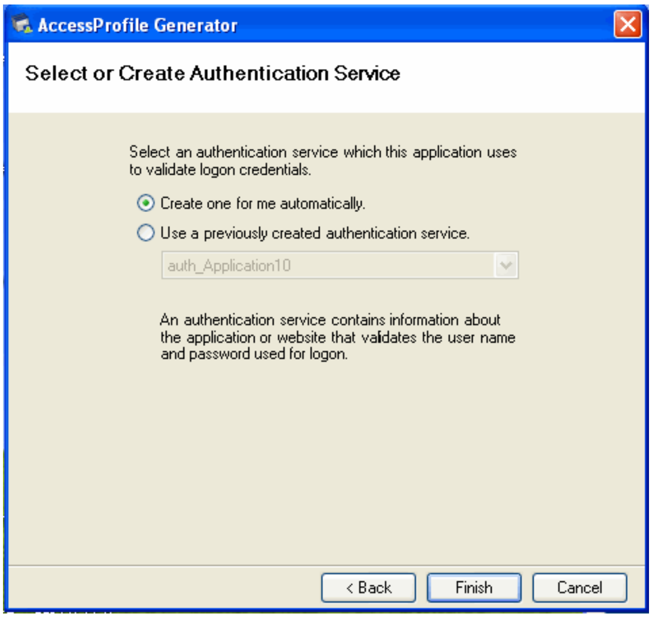
❿ Specify Actions for Logon, then click **Next** >.

⓫ Specify whether you want AccessStudio to identify the successful logon to avoid capturing incorrectly typed credentials.

If you selected **Yes**, provide a unique case-sensitive screen text for identification.

Click **Next >** to proceed.

⓬ Select or create authentication service.



AccessProfile Generator

Select or Create Authentication Service

Select an authentication service which this application uses to validate logon credentials.

(●) Create one for me automatically.

( ) Use a previously created authentication service.

auth_Application10

An authentication service contains information about the application or website that validates the user name and password used for logon.

Select or Create Authentication Service dialog box.

⓭ Click **Finish** to return to the AccessStudio user interface.

Test all the AccessProfiles before uploading to IMS.

⓮ Upload the AccessProfile to IMS Server to activate it. In the left pane, right-click on the AccessProfile, and select **Upload to IMS**. Click **Yes** when the **IMS Upload Confirmation** appears.

Another message box appears, indicating the success or failure of the upload.

Using AccessProfile Generator for Mainframe or cursor-based applications

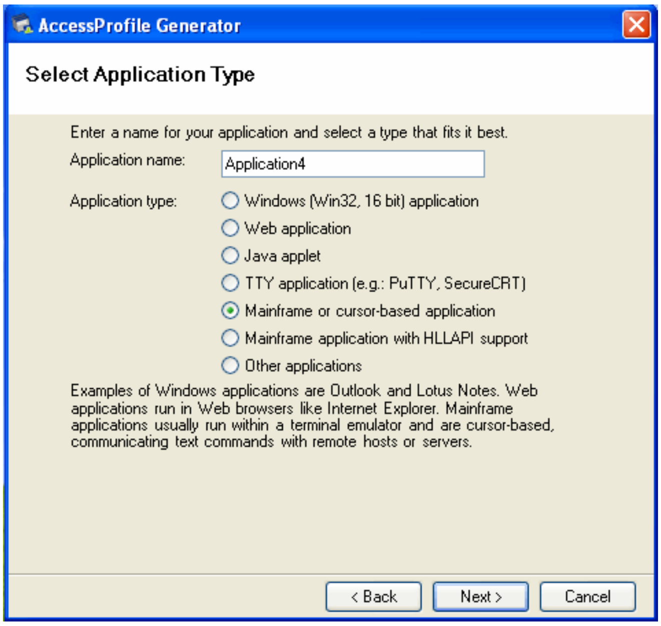
Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

To create AccessProfiles using the AccessProfile Generator for Mainframe or cursor-based applications:

❶ See Using the AccessProfile Generator for Windows applications for the first two steps on starting the AccessProfile Generator,

❷ Open the application logon screen that will require an AccessProfile.

❸ Once you have opened the application screen or Web page, click **Next >** to proceed.



Enter a name for your application and select a type that fits it best.

Application name: Application4

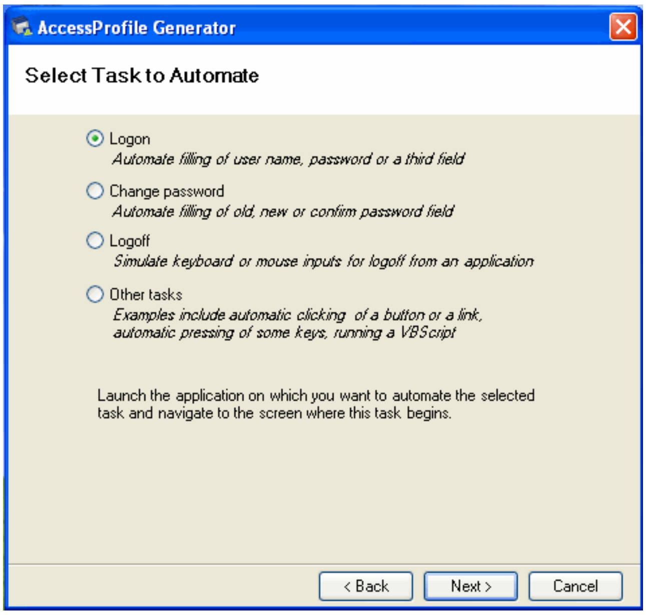
Application type:
- Windows (Win32, 16 bit) application
- Web application
- Java applet
- TTY application (e.g.: PuTTY, SecureCRT)
- [✓ Mainframe or cursor-based application]
- Mainframe application with HLLAPI support
- Other applications

Examples of Windows applications are Outlook and Lotus Notes. Web applications run in Web browsers like Internet Explorer. Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

Select Mainframe or cursor-based application.

❹ Enter a unique name for the application and select **Mainframe or cursor-based application** as the application type. Click **Next >** to proceed.

❺ Specify the task you want to automate by marking the appropriate radio button. Click **Next** > to proceed.



AccessProfile Generator

Select Task to Automate

(●) Logon
Automate filling of user name, password or a third field

( ) Change password
Automate filling of old, new or confirm password field

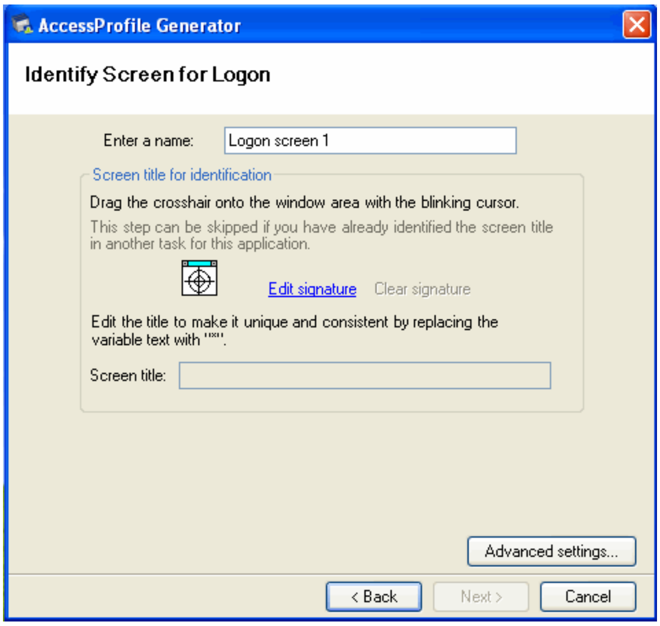
( ) Logoff
Simulate keyboard or mouse inputs for logoff from an application

( ) Other tasks
Examples include automatic clicking of a button or a link, automatic pressing of some keys, running a VBScript

Launch the application on which you want to automate the selected task and navigate to the screen where this task begins.

Select a task to automate.

❻ Enter a unique name for the screen or Web page to capture.



AccessProfile Generator

Identify Screen for Logon

Enter a name: Logon screen 1

Screen title for identification

Drag the crosshair onto the window area with the blinking cursor.

This step can be skipped if you have already identified the screen title in another task for this application.

Edit the title to make it unique and consistent by replacing the variable text with "*".

Screen title:

Enter a unique name for the application data to capture.

❼ Based on your selected task, capture identification information for the fields on the application screen.

The fields available for each task to be automated will vary.

8. Click on a crosshair icon from the AccessProfile Generator, then drag and drop the icon on the matching field in the application screen or Web page. As you drag the crosshair icon to the application, AccessProfile Generator marks the field or button that can be captured. Once the crosshair icon is positioned over the field, release the mouse button.

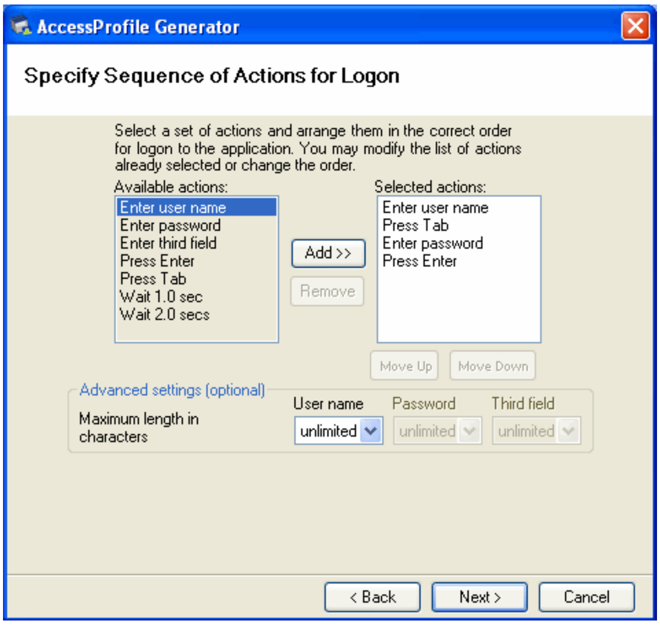
   If the field was captured successfully, the **Clear** option is activated. The screen title field retrieves its default screen name from the application. Click **Clear** to undo the capture.

9. After capturing all the application fields, click **Next >** to proceed.

10. In the **Identify Screen for Logon** dialog box, provide case-sensitive text string/s that can accurately identify the screen. Click **Add**.

11. Click **Next >** to proceed.

12. Specify **Sequence of Actions for Logon**, then click **Next >**.



AccessProfile Generator

Specify Sequence of Actions for Logon

Select a set of actions and arrange them in the correct order for logon to the application. You may modify the list of actions already selected or change the order.

Available actions:
- Enter user name
- Enter password
- Enter third field
- Press Enter
- Press Tab
- Wait 1.0 sec
- Wait 2.0 secs

Selected actions:
- Enter user name
- Press Tab
- Enter password
- Press Enter

Advanced settings (optional)

	User name	Password	Third field
Maximum length in characters	unlimited	unlimited	unlimited

Select available actions from the left pane and click Add>>.

⓭ Specify whether you want AccessStudio to identify the successful completion of the task (for example: logon, logoff, change password, etc.).

The options are **No** (no success screen or message appears), **Yes, identify the screen that appears** (capture the success screen of the application), or **Yes, simply detect closure** (shows no success screen but shows a message).

If you selected **Yes, identify the screen that appears**, drag and drop the crosshair icon on the success application screen or Web page. Once the crosshair icon is positioned over the screen or Web page, release the mouse button. Based on the captured item, you can also enter the screen title or text.

Click **Next >** to proceed.

⓮ Select or create authentication service.

⓯ Click **Finish** to return to the AccessStudio user interface.

Test all the AccessProfiles before uploading to IMS.

⓰ Upload the AccessProfile to IMS Server to activate it. In the left pane, right-click on the AccessProfile, and select **Upload to IMS**. Click **Yes** when the **IMS Upload Confirmation** appears.

Another message box appears, indicating the success or failure of the upload.

Using the AccessProfile Generator for Mainframe application with HLLAPI support

HLLAPI is an abbreviation for High Level Language API, which is a standard to access legacy information and applications housed on IBM and Unisys mainframes, AS/400, UNIX/VMS, etc. Examples of applications that provide HLLAPI are Attachmate Extra, Reflection, and IBM ISeries. Terminal Emulators are typically used to connect to these legacy applications.

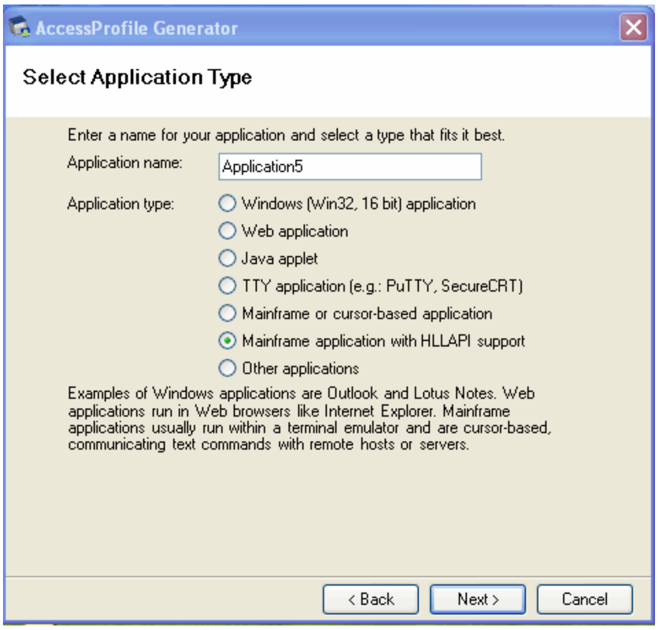
To create AccessProfiles using the AccessProfile Generator for Mainframe application with HLLAPI support:

❶ See Using the AccessProfile Generator for Windows applications for the first three steps on starting the AccessProfile Generator.

❷ Open the application logon screen that will require an AccessProfile.

❸ Once you have opened the application screen or Web page, click **Next >** to proceed.

❹ Enter a unique name for the application and select **Mainframe application with HLLAPI support** as the application type. Click **Next** > to proceed.



Select Application Type

Enter a name for your application and select a type that fits it best.

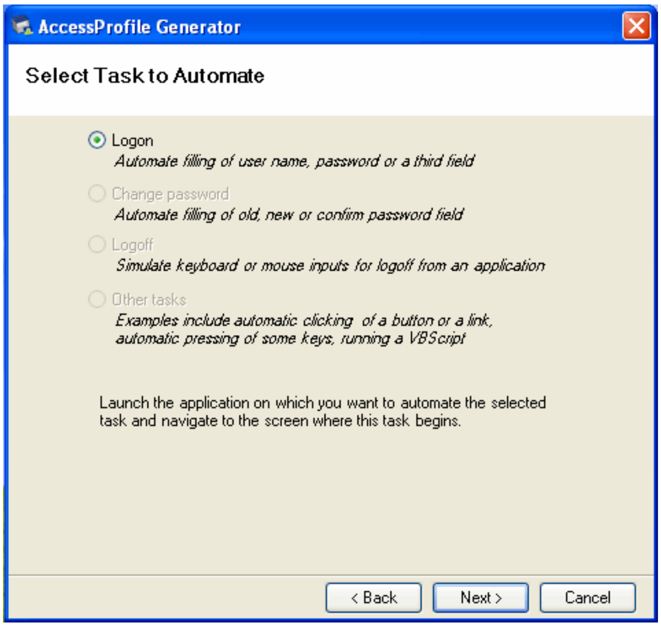
Application name: Application5

Application type:
- ○ Windows (Win32, 16 bit) application
- ○ Web application
- ○ Java applet
- ○ TTY application (e.g.: PuTTY, SecureCRT)
- ○ Mainframe or cursor-based application
- ● Mainframe application with HLLAPI support
- ○ Other applications

Examples of Windows applications are Outlook and Lotus Notes. Web applications run in Web browsers like Internet Explorer. Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

Select Mainframe application with HLLAPI support

❺ In the **Select Task to Automate** screen, **Logon** is the only option activated. Click **Next** > to proceed.



Select Task to Automate

- ● Logon
  Automate filling of user name, password or a third field
- ○ Change password
  Automate filling of old, new or confirm password field
- ○ Logoff
  Simulate keyboard or mouse inputs for logoff from an application
- ○ Other tasks
  Examples include automatic clicking of a button or a link, automatic pressing of some keys, running a VBScript

Launch the application on which you want to automate the selected task and navigate to the screen where this task begins.

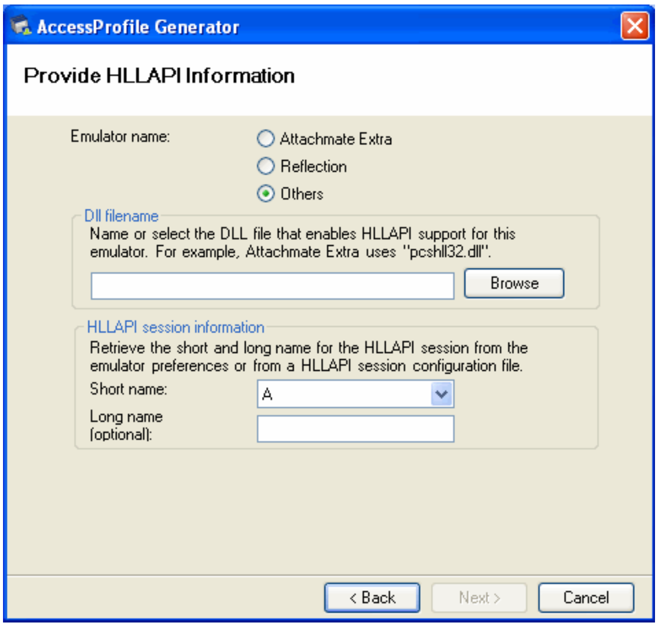
Logon is selected by default.

❻ Provide the necessary HLLAPI information, then click **Next >**.

Determine and configure the short name and long name of the host session on the legacy application.

All installations of Terminal Emulators must have the following settings. Configuration for each application varies. Refer to the documentation of the specific application for configuration information.

- **Session Short Name**: Unique name to identify a host session. Single letter from A to Z. This is mandatory.
- **Session Long Name**: Alternate Name to identify a host session, up to 8 characters.
- **HLLAPI DLL**: Dynamic Linked Library for HLLAPI functionality provided by the Terminal Emulator Application.


AccessProfile Generator

Provide HLLAPI Information

Emulator name: ( ) Attachmate Extra ( ) Reflection [✓ Others]

Dll filename

Name or select the DLL file that enables HLLAPI support for this emulator. For example, Attachmate Extra uses "pcshll32.dll".

HLLAPI session information

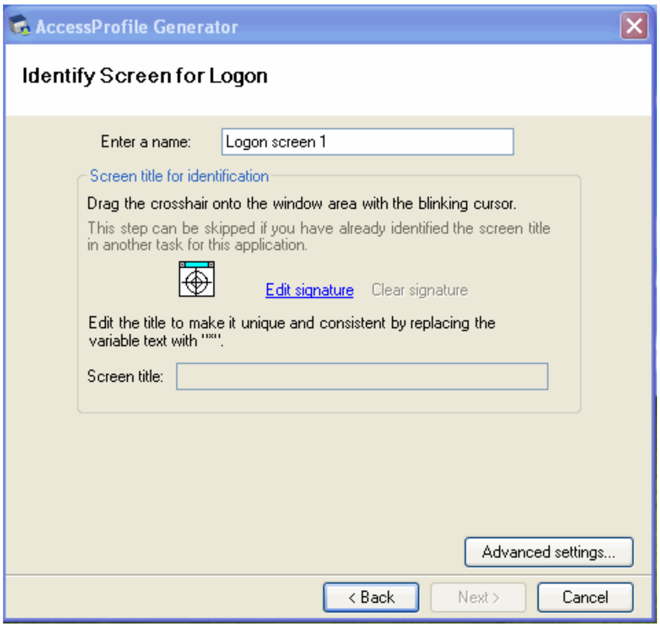
Retrieve the short and long name for the HLLAPI session from the emulator preferences or from a HLLAPI session configuration file.

Short name: A

Long name (optional):

Fill out the necessary HLLAPI information and click Next >.

❼ In the **Identify Screen for Logon** dialog box, enter a unique name for the screen or Web page to capture.


AccessProfile Generator

Identify Screen for Logon

Enter a name: Logon screen 1

Screen title for identification

Drag the crosshair onto the window area with the blinking cursor.

This step can be skipped if you have already identified the screen title in another task for this application.

Edit signature Clear signature

Edit the title to make it unique and consistent by replacing the variable text with "*".

Screen title:

Enter a unique name for the application data you need to capture.

❽ Based on your selected task, capture identification information for the fields on the application screen.

The fields available for each task to be automated will vary.

❾ Click on a crosshair icon from the AccessProfile Generator, then drag and drop the icon on the matching field in the application screen or Web page. As you drag the crosshair icon to the application, AccessProfile Generator marks the field or button that can be captured. Once the crosshair icon is positioned over the field, release the mouse button.

If the field was captured successfully, the **Clear** option is activated. The screen title field retrieves its default screen name from the application. Click **Clear** to undo the capture.

❿ After capturing all the application fields, click **Next** > to proceed.

⓫ In the **Identify Screen for Logon** dialog box, provide case-sensitive text string/s that can accurately identify the screen. Click **Add**.

⓬ Click **Next** > to proceed.

⓭ Specify **Sequence of Actions for Logon**, then click **Next** >.

⓮ Specify whether you want AccessStudio to identify the successful completion of the task (for example: logon, logoff, change password, etc.).

The options are **No, continue without identifying successful logon**, and **Yes, find the screen that appears upon successful logon**.

If you selected **Yes, find the screen that appears**, key in one or more case-sensitive text strings that appear on the screen after successful logon. Then click **Add**.

Click **Next >** to proceed.

⓯ Select or create authentication service.

⓰ Click **Finish** to return to the AccessStudio user interface.

Test all the AccessProfiles before uploading to IMS.

⓱ Upload the AccessProfile to IMS Server to activate it. In the left pane, right-click on the AccessProfile, and select **Upload to IMS**. Click **Yes** when the **IMS Upload Confirmation** appears.

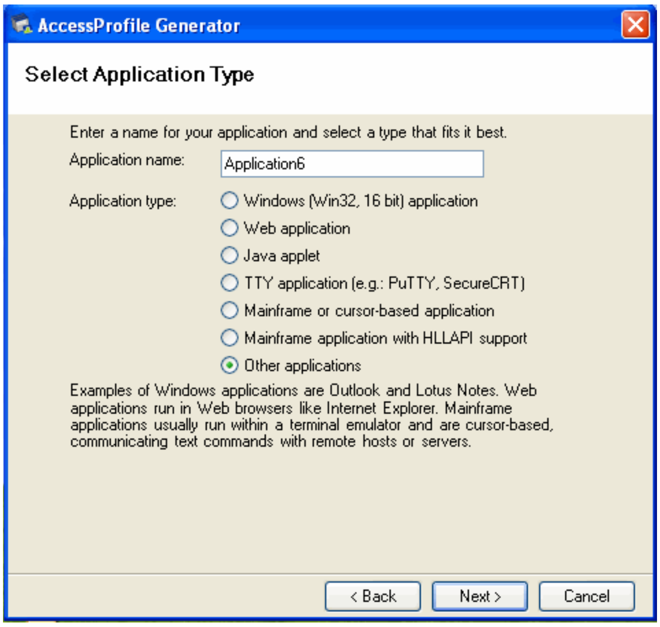
Another message box appears, indicating the success or failure of the upload.

Using the AccessProfile Generator for other applications

❶ See Using the AccessProfile Generator for Windows applications for the first three steps on starting the AccessProfile Generator.

❷ Open the application logon screen that will require an AccessProfile.

❸ Once you have opened the application screen or Web page, click **Next >** to proceed.


AccessProfile Generator

Select Application Type

Enter a name for your application and select a type that fits it best.

Application name: Application6

Application type:
- ( ) Windows (Win32, 16 bit) application
- ( ) Web application
- ( ) Java applet
- ( ) TTY application (e.g.: PuTTY, SecureCRT)
- ( ) Mainframe or cursor-based application
- ( ) Mainframe application with HLLAPI support
- [✓ Other applications]

Examples of Windows applications are Outlook and Lotus Notes. Web applications run in Web browsers like Internet Explorer. Mainframe applications usually run within a terminal emulator and are cursor-based, communicating text commands with remote hosts or servers.

Select Other applications

❹ Enter a unique name for the application and select **Other application** as the application type. Click **Next >** to proceed.

❺ Specify the task you want to automate by marking the appropriate radio button. Click **Next >** to proceed. The **Logon** option is selected by default.

❻ Enter a unique name for the screen or Web page to capture in the Identity Screen for Logon dialog box.

❼ Based on your selected task, capture identification information for the fields on the application screen.

The fields available for each task to be automated will vary.

❽ Click on a crosshair icon from the AccessProfile Generator, then drag and drop the icon on the matching field in the application screen or Web page. As you drag the crosshair icon to the application, AccessProfile Generator marks the field or button that can be captured. Once the crosshair icon is positioned over the field, release the mouse button.

If the field was captured successfully, the **Clear** option is activated. The screen title field retrieves its default screen name from the application. Click **Clear** to undo the capture.

❾ After capturing all the application fields, click **Next >** to proceed.

⑩ In the **Identify Screen for Logon** dialog box, you can also add unique screen text for identification. Click **Next** > to proceed.

⑪ Specify **Sequence of Actions for Logon**, then click **Next** >.

⑫ Select or create authentication service.

⑬ Click **Finish** to return to the AccessStudio user interface.

Test all the AccessProfiles before uploading to IMS.

⑭ Upload the AccessProfile to IMS Server to activate it. In the left pane, right-click on the AccessProfile, and select **Upload to IMS**. Click **Yes** when the **IMS Upload Confirmation** appears.

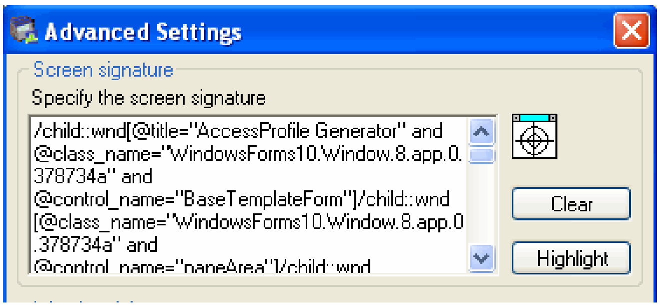
Another message box appears, indicating the success or failure of the upload.

Editing standard AccessProfile

To edit standard AccessProfile:

❶ Click **Advanced settings...** in the **Identify Screen and Fields for Logon** window.

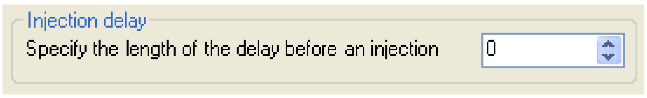
❷ Edit the signature in the **Screen signature** field.



Specify the screen signature

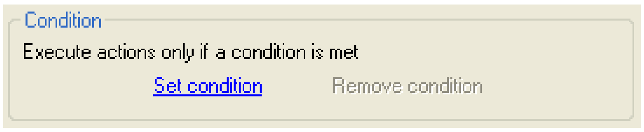
```
/child::wnd[@title="AccessProfile Generator" and
@class_name="WindowsForms10.Window.8.app.0.
378734a" and
@control_name="BaseTemplateForm"]/child::wnd
[@class_name="WindowsForms10.Window.8.app.0
.378734a" and
@control_name="paneArea"]/child::wnd
```

❸ The auto-fill delay is set to **0** by default. If auto-fill fails, specify the length of the delay before an auto-fill.



Specify the length of the delay before an injection: 0

❹ Click the **Set condition** link if you want to set conditions for customized triggers.


Condition

Execute actions only if a condition is met

❺ Click **OK**.

Advanced AccessProfiles

The AccessProfile Generator automatically generates AccessProfile but it does not correspond to a state-engine script or advanced AccessProfile. To convert this AccessProfile into the state-engine script, select the simple_sso_support node from the left pane and right-click. Choose **Convert to State Engine Support** from the dropdown menu and the simple_sso_support becomes an advanced AccessProfile.

Another way to create advanced AccessProfiles is by using the Form Editor.

This chapter covers the following topics:

- Creating advanced AccessProfiles using the Form Editor
- Editing advanced AccessProfiles

Creating advanced AccessProfiles using the Form Editor

To create an Advanced AccessProfile (State Engine SSO Support) using the Form Editor:

1. Add an AccessProfile.
2. Specify the application type.

Add support for TTY or Java based on the application type if necessary.

3. Add states.
4. Add triggers.
5. Add actions.

❻ Capture identification information for the logon fields on the application screen.

❼ Upload the AccessProfile to IMS Server.

To create an Advanced AccessProfile using the Form Editor:

❶ Add an AccessProfile:

Right-click the **access_profiles** node in the left pane and select **Add AccessProfile**. An AccessProfile is automatically created for you with a default name of **sso_site** which you can modify using the right pane.

❷ Specify the application type:

Right-click the **sso_site** node, the go to *Add SSO Support >> State Engine SSO Support*. State Engine SSO Support in AccessStudio indicates an Advanced AccessProfile. A new sub-node with the default name **state_engine_sso_support** is created under the **sso_site** node.

Add support for TTY or Java based on the application type, if necessary.

❸ Select the application object:

Click the **site_info** node. If the application object is not yet created, see Managing Application Objects, Creating an application object for instructions to create it.

❹ Add states:

The **state_engine_sso_support** node has a **states** sub-node. Right-click the **states** sub-node and select **Add State**. A new sub-node is created with the default name **new_state**. Specify the state information in the Form Editor.

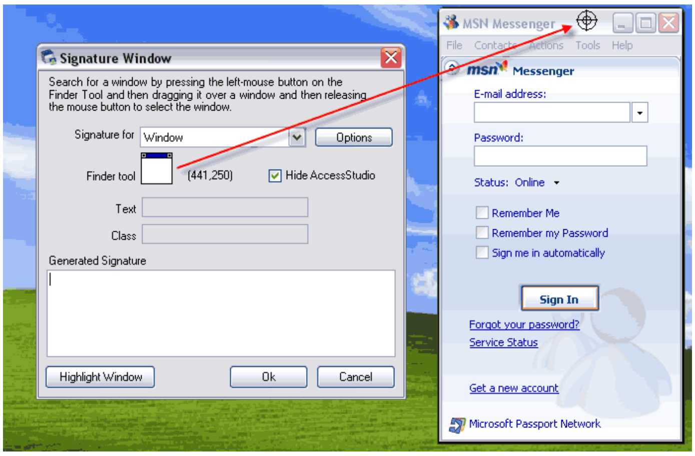
❺ Add triggers:

The **new_state** node has a **triggers** subnode. Right-click this node, point to **Add Triggers** and select the desired trigger from the list. Specify the trigger information in the Form Editor. For some triggers you may need to capture the signature of an associated window. A signature is unique identification information for any window or field.

To capture a signature:

1. Click the crosshair icon next to a field for capturing a signature to display the Signature window.

2. Click the **Finder tool** crosshair, drag and drop it on the field or window of the application for which you are configuring the AccessProfile. The signature is automatically generated and displayed in the **Generated Signature** text box.



Search for a window by pressing the left-mouse button on the Finder Tool and then dragging it over a window and then releasing the mouse button to select the window.

Signature for: Window

Finder tool (441,250)

[✓] Hide AccessStudio

Status: Online

❺ Add actions:

Once you specify the trigger information, you can now associate actions with the triggers. Right-click the **actions** node, point to **Add Action** and select the desired action from the list. Specify the action information in the Form Editor.

If your desired action is not listed, you can script your own action. See Configuring customized actions section for more details. For some actions you may need to capture the signature of an associated window. Follow instructions in step 4 to capture a window signature.

❻ Capture identification information for the logon fields on the application screen:

You can specify credential information for capture via sso-items for some actions. AccessAgent captures or auto-fills credential information from fields on a screen called sso-items.

For example, the user name or password field on a window. Sso-items correspond to info-controls in a Standard AccessProfile. You can also configure sso-items to include the format of information to be captured. For instance, if you are creating an AccessProfile for MSN Messenger, and all users in your organization are required to have Hotmail accounts, you can create two actions for capturing and auto-filling the information. In the first action you can specify that when capturing information from an MSN Messenger, AccessAgent should ignore the '@hotmail.com' and capture only the user name. Then, in the second action you can specify that AccessAgent should append a captured MSN Messenger user name with '@hotmail.com' for all users.

7. Upload the AccessProfile to IMS Server

   Once you have created the AccessProfile you can upload it to IMS Server to which your AccessAgent is connected. This will activate the AccessProfile. To upload an AccessProfile to the IMS Server, right-click the AccessProfile and click **Upload to IMS**. A message will appear indicating the success or failure of the upload.

Configuring customized triggers

You can create customized triggers in AccessStudio if the desired trigger is not available in the pre-defined trigger options. Customized triggers can be created by adding different test conditions.

A condition enables you to specify extra criteria that must be met before the trigger associated with it fires. The condition can be checking a property value against a string, or a JScript or VBScript view. It meets the condition if the function returns true. You can specify multiple conditions for the same trigger. In such a case, all conditions have to be satisfied before the trigger fires.

Examples of test conditions: test account data bag, test property and test with script.

To configure customized triggers:

1. Right-click on each trigger.

2. Select **Add Conditions**.

3. Add a **test_with_script** condition.

   The **test_with_script** condition enables you to specify a script to use for the test. The condition will be satisfied only if the script is evaluated as true.

You can access all the local properties in the script.

Example:

A sample VBScript to perform a condition test is given below:

The test condition must be specified in a function called test, which returns a boolean value. Script to test if the property total is less than 10.

Function test:

```
Set pc=runtime.GetPropertiesContainer()

b=pc.GetPropValue("total")

if b="" then

    test=True

else

    c=CInt(b)

    if c<10 then

         test=True

    else

         test=False

    end if

end if

end function
```

You can write scripts in VBScript or JavaScript.

Configuring customized actions

Actions available in AccessStudio are built-in. AccessStudio also enables you to create customized actions if your desired action is not available in the ones provided in AccessStudio. Customized actions can be created by writing scripts in any language supported by your computer, for example VBScript or JScript. AccessStudio provides support for these scripts to interact with AccessAgent.

To create a customized action:

1. Right-click the **actions** node, point to *Add Action >> Advanced*, and select **run_script_action**. The corresponding fields will display in the Form Editor.
2. Enter the script code in the **Script** field. The action will be active once you upload the AccessProfile to IMS Server.

The following is an example of a possible requirement and configuration for a customized action. Consider that you may want to capture, and save the user name in AccessAgent only in an upper case format. However, a user may type the user name in any format. This would require you to create a customized action to ensure that the user name is always captured in the upper case. A typical user credential capture would have only two actions - capture and save.

To convert a captured user name into uppercase:

1. Capture user name in account data bag.
2. Transfer the captured user name into a property item.
3. Convert the user name into upper case using the script.
4. Transfer the converted user name back to the account data bag.
5. Save.

To do this using AccessStudio:

1. Create a wnd_command_bn_click_trigger and specify the required information.
2. Under this trigger, create an acc_data_capture_action. Specify the ID of the account data bag which you are using to capture the user name.
3. Create a data_transfer_action.
4. Create a data transfer item under the action. Next, under the From node, right-click and add acc_data_bag_item. Select aditi_cuser as the data transfer item type, and specify the ID of the account data bag you are using. Under the To node right-click and add the property_store_item as the data transfer item. Specify usernamelower as an ID for this item.
5. Create a run_script_action. Change the language setting to JScript and enter the following code in the Script field:

   ```
   var pc=runtime.GetPropertiesContainer();
   var b=pc.GetPropValue("usernamelower");
   pc.SetPropValue("usernamelower",b.toUpperCase());
   ```

6. Next, create another data transfer action. Under this action create a data transfer item. Under the From node, create a property_store_item data transfer item. Specify usernamelower as an ID for this item. Under the To node, add acc_data_bag_item. Select aditi_cuser as the data transfer item type, and specify the ID of the account data bag you are using.
7. Finally create an acc_data_save_action action and specify the ID of the account data bag you are using.

Editing advanced AccessProfiles

There are many ways to edit your advanced AccessProfiles. You can edit AccessProfiles from any of the following:

- Form Editor tab
- XML tab

Editing advanced AccessProfile from the Form Editor tab

To edit from the Form Editor tab:

❶ Select the node you need to edit from the left pane. This automatically displays the Form Editor tab on the right pane.

❷ Click Edit from the Signatures identifying web-page or exe where this profile is to be loaded field to start editing. This displays the Signature window.

❸ Select the appropriate option from the Signature for dropdown menu.

❹ Edit the signature from the Generated Signature field.

❺ Mark the checkbox based on what you need to generate.

❻ Click Ok to return to the Form Editor tab.

Editing advanced AccessProfile from the XML Editor tab

To edit from the XML Editor tab:

❶ Click Edit from the Signatures identifying web-page or exe where this profile is to be loaded field. This displays the Signature window.

❷ Select the appropriate option from the Signature for dropdown menu.

❸ Edit the signature from the Generated Signature field.

Managing Authentication Services

Most applications require validation of logon information by a verification entity. In AccessStudio, a reference is created to these entitites through Authentication Services. AccessProfiles associated with the same authentication service belongs to the same verification entity. Changes made to the logon information in one AccessProfile are reflected across all others associated with the authentication service. This concept is explained in the Authentication service section.

All these applications must share the same set of credentials. If you associate applications with different sets of credentials with the same authentication service, it will result in an error for the specific AccessProfiles. User credentials are stored in the Encentuate Wallet according to the authentication service and *not* the application.

This section will be useful to those using the Form Editor or XML Editor to create AccessProfiles.

This chapter covers the following topics:

- Associating authentication services with AccessProfiles
- Creating an authentication service
- Modifying an authentication service
- Managing authentication service groups and group links

Associating authentication services with AccessProfiles

You can define authentication services in AccessStudio using the **Authentication Services** function in the **View** menu. At a minimum, you need to provide an ID and a display name for the authentication service. Additional information may be specified depending on your requirements. Hence, authentication services can be associated with AccessProfiles in two ways - direct and indirect.

Direct auth-info

Direct auth-info is a direct reference to an existing authentication service configured using the **Authentication Services** function in AccessStudio. When you configure an authentication service to be used as a direct reference, you only need to specify the authentication service ID and display name. This ID is displayed in a drop-down list when configuring an AccessProfile. This is the direct reference. In a majority of the cases a direct auth-info reference is sufficient.

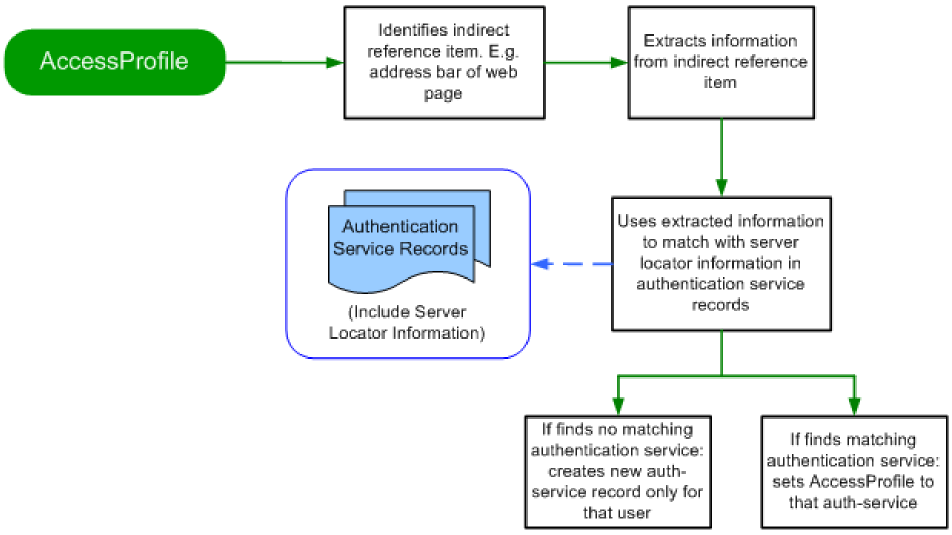
Indirect auth-info

Indirect auth-info is an indirect reference to an existing authentication service. When you configure an authentication service to be used as an indirect reference, in addition to the ID and display name, you must provide information about the server locators. Server locators help identify the entity that verifies the user's logon information.

An example of a server locator for a Website is xyz.com. This server locator information will be used when you specify information about the indirect reference. Indirect references are made to controls, elements, or windows of applications from which information is extracted and matched with the information specified for a server locator.

Once a match is made the authentication service is determined and associated with the application. In addition you can specify a regular expression to further refine the information extracted from the indirect reference item. Indirect auth-info references are useful in cases where you are unsure of what authentication service to use in your AccessProfiles.

The indirect reference process is illustrated in the following diagram.


AccessProfile
Identifies indirect reference item. E.g. address bar of web page
Extracts information from indirect reference item
Uses extracted information to match with server locator information in authentication service records
Authentication Service Records
(Include Server Locator Information)
If finds no matching authentication service: creates new auth-service record only for that user
If finds matching authentication service: sets AccessProfile to that auth-service

The diagram illustrates that once an AccessProfile is executed, it identifies the indirect reference item specified in the AccessProfile. Next, it extracts information from the reference item.

For example, it can extract information from the address bar of a Web page. It then tries to match the extracted information with that specified for the server locators of existing authentication service records on IMS Server.

If it finds a matching record, it sets the authentication service to the ID of the matching authentication service record. If it does not find a matching authentication service record, it creates a new authentication service record only for that user and creates an association with this record. Note that this newly-created authentication service will not be available for any other user or at IMS Server for further association with new AccessProfiles.

Creating an authentication service

To create a new authentication service:

1. Go to *View >> Authentication Services*.

2. Right-click on the **authentication_services** node and select **Add Authentication Service**. A new sub-node is created for you, and the corresponding fields are displayed on the Form Editor.

3. Enter a unique name to identify the authentication service in the **Id** field.

4. Specify a name for the authentication service to be displayed in AccessAgent's Wallet in the **Display Name** field. The ID and display name are mandatory information that are sufficient to create a direct auth-info reference.

5. Enter a short description for this authentication service in the **Description** field. This is optional.

6. You can also specify the account data template to use for the authentication service. An account data template defines the structure of the account data, that is, user credentials.

   The default account data template ID is **adt_ciuser_cspwd**. This indicates the nature of user credentials as a case-insensitive user name and a case-sensitive password.

7. Specify server locator information. This is mandatory if you want to use an indirect auth-info reference. Specify server locator information for the applications you want to associate with this authentication service in the text box beside the + button. For example enter www.xyz.abcd.com. Click the + button.

   You can add multiple server locators under one authentication service. This is useful in case you have multiple applications which use the same authentication service but have different server locators.

For example, you can have a domain named **abcd.com**, and its sub-domains use the same authentication. So you can specify server locators for each of the services running on the sub-domains like **http://123.abcd.com**, **http://456.abcd.com**, and so on.

You can also specify separate server locators for auto-fill of user credentials and that for credential capture. Clear the selection in the **Use same server locators for injection and capture** field to display separate server locator fields for auto-fill and capture.

Modifying an authentication service

To modify authentication services, select the authentication service node from the left pane and modify the information in the Form Editor. However, if you are modifying the **Id** field, ensure that no AccessProfiles are currently associated with the authentication service as it can result in errors with the AccessProfiles.

Managing authentication service groups and group links

Associating AccessProfiles with independent authentication services is usually sufficient. However, in rare cases even the user interface of an application cannot provide a clue to identify the authentication service. In such cases you can create an Authentication Service Group, and associate multiple authentication services with this group.

You can configure an authentication service group using the **Authentication Service Groups** function in AccessStudio. Once you create a group, you can link authentication services with the group using the **Authentication Service Group Links** function. You can then associate this group with an advanced AccessProfile. The group associations are displayed to the user as logon options when the user accesses the application.

You can associate authentication service group only with advanced AccessProfiles.

The following is an example of a scenario where you might need to create authentication service groups and group links.

Assume that you have an application named ENC. ENC has two authentication services that validate user credentials - **auth_portal** and **auth_enterprise**. You have no way to determine which authentication service must be used for logon.

Hence you create an authentication service group named **authgroup_ENC** and create a link between the group and the two authentication services **auth_portal** and **auth_enterprise**. Next, you associate this group with an advanced AccessProfile you have created for the ENC application.

Thus, when the user accesses the ENC application, he or she will be prompted to select the authentication service to use for logon.

Creating an authentication service group and authentication service group link

To create an authentication service group:

❶ Go to *View >> Advanced Data >> Authentication Service Groups*. This displays the **authenticator_groups** node in the left pane. Existing groups are displayed as sub-nodes under this node.

❷ Right-click the **authenticator_groups** node and click **Add Authentication Service Group**. A new sub-node is created for you, and the corresponding fields are displayed on the Form Editor.

❸ Enter a unique identification name for the group in the **Id** field.

❹ You can also create an indirect reference for the authentication service group as you did for an authentication service by specifying the server locator information.

❺ Next, create a link between the group and relevant authentication services. Go to *View >> Advanced Data >> Authentication Service Group Links*. This displays the **authenticator_group_links** node in the left pane. Existing groups are displayed as sub-nodes under this node.

❻ Right-click the **authenticator_group_links** node and select **Add Authentication Service Group Link**. A new sub-node is created, and the corresponding fields are displayed on the Form Editor.

❼ Select the authentication service group link ID, and enter the **Authentication service group id** to be linked with the authentication service.

❽ Enter the **Authentication service id** of the authentication service to be linked with the authentication service group.

Modifying an authentication service group

To modify an authentication service group, access the Authentication Service Groups function, select the authentication service group node from the left pane and modify the information in the Form Editor. However, if you are modifying the **Id** field, ensure that no authentication services currently linked with the authentication service group as it can result in errors with the AccessProfiles.

Modifying an authentication service group link

To modify an authentication service group link, access the Authentication Service Group Links function, select the authentication service group link node from the left pane and modify the information in the Form Editor.

Managing Application Objects

An application object in AccessStudio is a logical representation of a set of executable files (**.EXE**) or Web pages. It provides you with tighter control to apply policies on a group of AccessProfiles. Every AccessProfile must be associated with an application object. Many AccessProfiles can be associated with the same application object.

This chapter covers the following topics:

- Creating an application object
- Modifying an application object

Creating an application object

To create a application object:

1. Go to *View >> Applications.*
2. Right-click the **applications** node and select **Add Application**. This displays the corresponding fields on the Form Editor.
3. Modify the application ID as per the conventions of your organization.
4. Enter the a name for the application object in the **Name** field. Ensure that the name is reflective of the group of exes or Web pages you are creating the application object to represent.
5. Enter description information in the **Description** field. This is an optional field.
6. Once you have created the application object, upload it to IMS Server to which your AccessAgent is connected. This makes the application object available for all users. To upload to IMS Server, right-click the application object and click **Upload to IMS**. A message appears indicating the success or failure of the upload.

Modifying an application object

To modify an application object in the future, select the application object in the left pane and edit the details. If you are modifying the application object ID, ensure that no AccessProfile is associated with the application object, or it will result in an error with the associated AccessProfiles.

Account Data Items and Templates

Account data represents a user's logon information in AccessStudio. This consists of the user name and password. The account data for an AccessProfile is stored in a specific format defined in the account data templates.

Account data templates includes individual account data items. The properties of these items are defined in the account data item templates. These templates are accessible in AccessStudio through the account data templates and account data item template functions. The templates are predefined in AccessStudio. You can view using the respective functions, but you cannot modify the templates.

This chapter covers the following topics:

- Account data item templates
- Account data templates
- Viewing account data item templates/account data templates

Account data item templates

An account data item template defines the properties of individual account data items. Account data, as explained earlier, refers to the user credentials required for logon. An account data item template defines whether the field entry is:

- a secret field that requires encryption
- case-sensitive or not

A user name field is usually not secret, and case insensitive. Password fields on the other hand are usually secret and case sensitive.

Account data items are identified through IDs in AccessStudio. Each ID is structured so that it instantly provides you with the properties of the account data item.

For example, **aditi_cspwd** can be broken into two parts: **aditi** and **cspwd**. **aditi** refers to 'account data item template information'. **cs** in the cspwd refers to 'case-insensitive', and **pwd** refers to 'password'. So, this account data item ID indicates that the account data item is a case-sensitive password field.

Account data templates

Account data templates define the format of account data to be stored for credentials captured using a specific AccessProfile. An account data template includes:

- Account data item templates
- Key identifier (identifies whether the account data item is key or non-key)

A key account data item indicates that it is a unique identifier for that set of account data. This means that information captured for this item cannot change. A non-key item on the other hand is considered non-constant and can be changed within that set of account data.

For example, a user name can be considered as the key item for the account data and the password the non-key. Thus, the password for that user name can change as many times as required without losing its identity as belonging to the account data set that carries the given user name.

If the user name changes, irrespective of whether the password is changed as well or not, the credentials will be treated as belonging to a different set of account data.

Account data templates are identified through IDs in AccessStudio. Each ID provides you with the properties of the account data items in the template. For example, **adt_csuser_cspwd** can be broken into three parts: **adt**, **csuser**, and **cspwd**. **adt** refers to 'account data template'. **csuser** refers to 'case-sensitive user name', and **cspwd** refers to 'case-sensitive password'. So, this account data template ID indicates that it contains two account data items - a case-sensitive user name and a case-sensitive password.

Account data templates can be associated with AccessProfiles and authentication services. Once associated, the account data template is used by the account data bag of each AccessProfile. An account data bag is a temporary data holder or cache that stores user credentials after their capture from the application screen, and before auto-fill.

The credentials are retrieved from Encentuate Wallet and stored in the account data bag before they are automatically inserted on the application screen fields. The user credentials are also stored in the account data bag after capture, before they are transferred to the user's Encentuate Wallet.

The account data template information is extracted either from the AccessProfile or the associated authentication service when an AccessProfile is executed. This template is then used to set the structure of the account data bag.

For example, the account data template adt_csuser_cspwd contains two account data items - a case-sensitive user name and a case-sensitive password. Thus, a structure is created for the account data bag which includes two slots. The first slot will be to capture a case-sensitive user name. The second slot will be to capture a case-sensitive password.

Next, store information in the correct slot from the corresponding fields. Then, associate an account data item template user name and password fields when you configure them in the AccessProfile.

In the given example, we would associate the aditi_csuser account data item template with the user name field of the application, and the aditi_cspwd account data item template with the password field. Thus, a simple match is conducted to identify data from which field belongs to which slot in the account data bag.

Viewing account data item templates/ account data templates

To view account data item templates/account data templates:

❶ Go to *View >> Advanced Data.*

❷ Select the function you want to view.

XPaths

AccessStudio extends the users' capability of editing the application's advanced AccessProfiles by using XPaths.

This chapter covers the following topics:

About XPath

XPath (XML Path Language) is a language that facilitates XML document navigation to select elements and attributes.

The XPath language has a hierarchical structure or a tree representation of a given XML document. It provides the ability to navigate around the tree of nodes and allows you to select nodes by a variety of criteria. From this tree, you may access the elements, attributes, and text nodes of your XML.

XPaths are used in signatures to identify the following:

- executables

  Example: `/child::exe[@exe_name="ypager.exe"]`

  Matches exe(s) with name YPager.exe.

- window elements (such as: edit control, buttons, and checkboxes)

  Example: `/child::wnd[@title="Login to YM"`

  Matches window(s) with title Login to YM and selects the descendent window(s) with class name matching the regx .*BUTTON.*. (# is for case-insensitive match)

- web pages

  Example: `/child::web[@domain="www.hotmail.com" and @proto-col="http"]`

  Matches Web pages from the URL with domain part equal to www.hot-mail.com and protocol equal to http.

- HTML elements (such as: submit buttons, input controls, etc.)

  Example: `/child::html/descendent::form/descen-dent::input[@tag_name="input" and @type="password"]`

  The first HTML refers to the head or the body, after that a form descendent is found and then a descendent of that form of tag-name input and type pass-word is searched for.

- Java window elements (such as: title, class name, window position, visibility status, size, etc.)

  Example: `/child::jwnd[@title="Login" and @class_name="MyJFrame"]`

  Matches window(s) with title "Login" and class name MyJFrame

These signatures can be edited in the AccessProfile Generator (for standard AccessProfiles), Form Editor and XML Editor (for advanced AccessProfiles).

Supported axes

Axis name	Description
child::	Child of current control subset [Default axis]
parent::	Parent of current control subset
descendent:: (or descendant::)	Descendent of current control subset
ancestor::	Ancestor of current control subset
self::	Current control subset
rhs::	Right Hand Side of control [1st control]
lhs::	Left Hand Side of control [1st control]
top::	Top of control [1st control]
btm::	Bottom of control [1st control]

Supported types

Name	Representation	Description
Executables	exe	To identify executables
Web pages	web	To identify Web pages/sites
Application (16 bit / Java / etc)	task	To identify 16-bits, Java and other hosted application
Windows	wnd	To identify windows
Java window	jwnd	To identify windows of Java application
HTML element	html	To identity HTML elements inside Web pages, including 'body' and 'head'
HTML Form	form	To identify HTML forms
HTML Input	input	To identify input fields in the HTML
HTML Frame	frame	To identify the HTML frame containing the document
HTML document	document	To identify the document containing the frameset or body and head
HTML anchor	anchor	To identify the anchor elements that has name or ID. (Anchors that don't have name or ID are not included).
HTML image	Image	To identify all the images inside the html document.

Available operators

Operator	Right hand side	Description
=	Numeric, String	Exact comparison, for strings the comparison is case-insensitive
~	String	Regex comparison, case-sensitive
#	String	Regex comparison, case-insensitive
!=	Numeric	Not equal.
!~	String	Not equal of regex comparison, case-sensitive
!#	String	Not equal of regex comparison, case-insensitive
&	Numeric	Binary AND
!&	Numeric	Not equal of binary AND
|	Numeric	Binary OR (NOT SUPPORTED IN THE CURRENT VERSION)
%%	Literal	The predefined literal between the two %s is translated to a numeric value
and	Logical	Logical and of two Booleans
or	Logical	Logical or of two Booleans

Significance of the root node ('/')

Type	Meaning
Executable	A hypothetical container of executables in the system
Web pages	A hypothetical container of all open Web pages in the system
HTML elements	A hypothetical container containing all the HTML of a Web page
Windows	A hypothetical container containing all the top level windows
HTML Form	Not allowed
HTML Input	Not allowed
HTML anchor	Not allowed
HTML image	Not allowed

No attributes are available for the root element, so no XPath can start with '/self::'

XPath for executables

XPaths for executables identify executables to determine whether or not an AccessProfile should be loaded for it. Available attributes are exe_name, company_name, file_version etc.

Available attributes

Attribute Name	Description
exe_name	The name of the executable
internal_name	The internal name
language	The language supported*
original_file_name	The original file name
product_name	The product name
product_version	The product version
file_version	The file version
wnd_title	The windows title of the top level window of application (16bits / Java / Others)
class_name	The class name of the top level window of application (16bits / Java / Others)
task_name	The task name of the application

The exe attribute names reflect the version information as seen on explorer for any exe.

Examples:

```
/child::exe[@exe_name="ypager.exe"]
```

- Matches exe(s) with name YPager.exe.

```
/child::task[@exe_name=”wowexec.exe” and
@task_name=”rumba.exe”]
```

- Matches 16bits application exe(s) with name “rumba.exe”

XPath for windows

Xpaths for window identify window elements such as: edit control, buttons, and checkboxes - a special form of windows which are also identified using this mechanism. The common attributes include title, class_name, ctrl_id etc.

Note that unlike Web and exe XPaths, Wnd XPath (and HTML XPath) support hierarchies, i.e. you can identify parent, ancestor, descendant and child relationships between windows.

Available attributes

Attribute Name	Type	Description
window_style	Numeric	Windows styles. Usually used along with bit-wise operators (&,|).
class_style	Numeric	Style of the class of window.
title	String	The title of the window as returned by the GetWindowText API. This would be the title, as seen by the user, of the top-level window; the text of the button etc. The value depends on the class of the window.
class_name	String	The class of the window.
ctrl_id	Numeric	Control id of the child window wrt to its parent.
xpos	Numeric	Horizontal position of the window wrt to the parent window.
ypos	Numeric	Vertical position of the window wrt to the parent window.
is_visible	Numeric (0/1)	Visibility status of the window.
size	Numeric	This is actually the area of the window (WIDHT*HEIGHT). Would be useful on windows that are not resizable.
window_ex_style	Numeric	Extended style of the window. Spy would be helpful in knowing which is extended style and which is not.
rel_xpos	Numeric	Taking into consideration the left-most position of all the windows, and numbering the windows from left to right starting from 1, the rel_xpos = n, represents the nth window. In case two windows share the same left x-pos, then they would get the same position, and the one afterwards would have the next position. So, for example, for 4 windows with left-top pos (2,5), (4,10), (4,15) and (10,4). The position assigned would be 1, 2, 2, 3.

Attribute Name	Type	Description
rel_ypos	Numeric	Taking into consideration the top-most position of all the windows, and numbering the windows from top to bottom starting from 1, the rel_ypos = n, represents the nth window. In case two windows share the same left y-pos, then they would get the same position, and the one afterwards would have the next position. So, for example, for 4 windows with left-top pos (2,5), (5,10), (4,10) and (10,15). The position assigned would be 1, 2, 2, 3.
file_path	String	Determine the file path of the executable that belongs to the window
file_name	String	Determine the file name of the executable that belongs to the window
control_name	String	Control name of the window control for .Net application

Examples:

```
/child::wnd[@title="Login to YM" and @class_name="YMWindow"]/
descendent::wnd[@class_name#".*BUTTON.*"]
```

- Matches window(s) with title Login to YM and selects the descendent window(s) with class name matching the regx .*BUTTON.*. (# is for case-insensitive match)
- When the element pointed to by the XPath is used to set or get text, the GetText function on the window is used, so the behavior depends on the class of the window (element).

XPath for Web pages

Xpaths for Web pages identify Web pages to determine whether or not an AccessProfile should be loaded for it. Available attributes are domain, protocol, port, etc.

Available attributes

Attribute Name	Type	Description
domain	String	Domain part of the URL (before the slash character and after the protocol identifier)
protocol	String	Protocol used
query_string	String	Query string in the URL

Attribute Name	Type	Description
path	String	Part of the URL after the domain and before the query string
port	Numeric	Port of connection
url	String	The complete URL

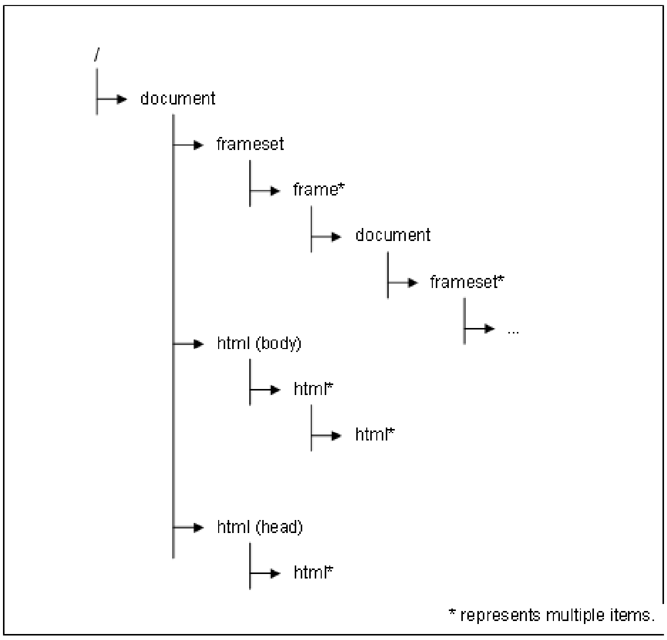
Example:

```
/child::web[@domain="www.hotmail.com" and @protocol="http"]
```

- Matches Web pages from the URL with domain part equal to www.hot-mail.com and protocol equal to http.

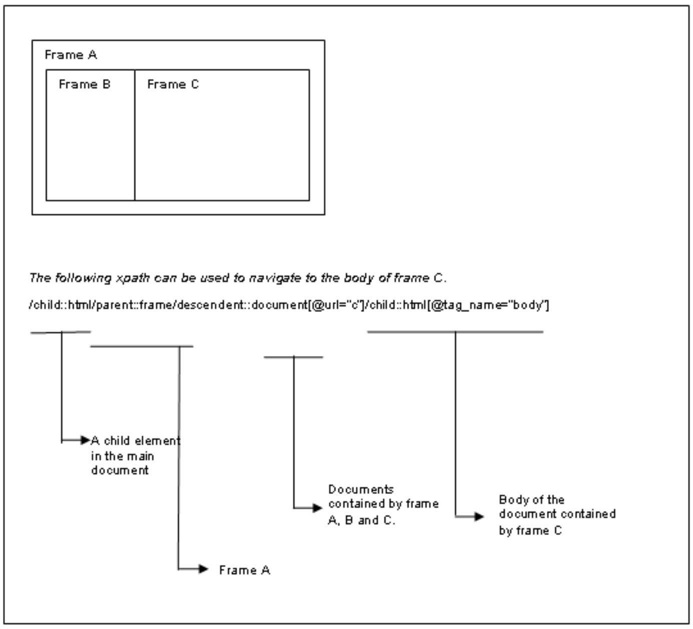
XPath for Web: conceptual hierarchy

XPath for Web is modeled to represent the following organization. Use the usual parent, child, ancestor and descendent axis to navigate from one node to another.



XPath for the Web conceptual hierarchy

Example of a frameset navigation


Frame A
Frame B
Frame C
A child element in the main document
Frame A
Documents contained by frame A, B and C.
Body of the document contained by frame C

The following xpath can be used to navigate to the body of frame C.

```
/child::html/parent::frame/descendent::document[@url="c"]/child::html[@tag_name="body"]
```

Frameset navigation example

XPath for HTML

XPaths for HTML identify HTML elements in a Web page such as: submit buttons, input controls, etc.

Available attributes [Common elements]

Attribute Name	Description
tag_name	Identify the type of HTML element. E.g. in the HTML snippet“<img src=”abd.gif”>” tag_name is “img”.
inner_text	Inner text of any HTML element. E.g. in the HTML snippet, ”<b>Login</b>”, Login is the inner text.
value	Value attribute of an HTML element. E.g. in the HTML snippet, ”<input value=”user name here”>”, ”user name here” is the value.
inner_html	Inner HTML of any HTML element. E.g. in the HTML snippet ”<p><b>login</b></p> the inner html of ‘p’ element is <b>login</b>.

There are also other miscellaneous attributes available for the element in the HTML DOM.

Value of the value *attribute of an element is returned using XPath to get the text of an HTML element.*

Examples [Common elements]

```
/child::html/descendent::form/
descendent::input[@tag_name="input" and @type="password"]
```

- The first HTML refers to the head or the body, after that, a form descendent is found and then a descendent of that form of tag-name input and type password is searched for.

```
/child::html/descendent::form/child::html[@tag_name=”table”]/
descendent::html[@tag_name=”b”]
/@inner_text
```

- Gets the inner text of a bold element that is inside a table that is inside a form.

```
/child::html/descendent::form/child::html[@tag_name=”table”]/
descendent::html[@tag_name=”b”]
```

- Points to the element. When text of this element is set the value attribute, if it exists, is set.

Available attributes [Form element]

Attribute Name	Description
form_name	The name of the form

There are also other miscellaneous attributes available for the 'form' element in the HTML DOM.

Examples [Form element]

```
/child::html/descendent::form[@tag_name="form" and @name="login"]
```

Available attributes [Input element]

Attribute Name	Description
input_name	The name of the input element
input_type	The type of the input element
className	Identifies the class attribute of the input element.
input_index	The index of the input element with respect to the form

descendent::input does not include input whose type *attribute is image. Use, though it is slow compared to descendent::input, descendent::html[@tag_name="input"] instead.*

There are also other miscellaneous attributes available for the 'form' element in the HTML DOM.

Examples [Input element]

```
/child::html/descendent::form/
descendent::input[@className="abc"]
```

- Will select all input element(s), which is inside any form, with class attribute equal to **abc**.

"/descendent::input" XPath does not select all the inputs from HTML document. To get all the input elements inside a document use "/descendent::form/descendent::input".

XPath for Java windows

XPaths for Java identify Java window elements such as: the title, class name, window position, visibility status, size, etc.

Available attributes

Attribute Name	Type	Description
title	String	The title of the java window. We support to get the title for the following java window class: java.awt.Frame, java.awt.Button, javax.swing.JFrame, javax.swing.JButton and all the other java classes which supports "getText()" method. This support also includes all the extended classes of the above mentioned classes.
class_name	String	The class name of the java window.
xpos	Numeric	Horizontal position of the window wrt to the parent window.
ypos	Numeric	Vertical position of the window wrt to the parent window.
is_visible	Numeric (0/1)	Visibility status of the window.
size	Numeric	This is actually the area of the window (WIDHT*HEIGHT). Would be useful on windows that are not resizable.
rel_xpos	Numeric	Taking into consideration the left-most position of all the windows, and numbering the windows from left to right starting from 1, the rel_xpos = n, represents the nth window. In case two windows share the same left x-pos, then they would get the same position, and the one afterwards would have the next position. So, for example, for 4 windows with left-top pos (2,5), (4,10), (4,15) and (10,4). The position assigned would be 1, 2, 2, 3.
rel_ypos	numeric	Taking into consideration the top-most position of all the windows, and numbering the windows from top to bottom starting from 1, the rel_ypos = n, represents the nth window. In case two windows share the same left y-pos, then they would get the same position, and the one afterwards would have the next position. So, for example, for 4 windows with left-top pos (2,5), (5,10), (4,10) and (10,15). The position assigned would be 1, 2, 2, 3.

Example:

```
/child::jwnd[@title="Login" and @class_name="MyJFrame"]
```

- Matches window(s) with title “Login” and class name MyJFrame

Grouping of Conditions

When using parenthesis to group conditions inside a predicate, all the groups should be inside their respective parenthesis temporarily including those that should not be included.

Example:

`/child::wnd[(@title=″a″ or @title=″b″) or @class_name=″c″]` will fail to evaluate because the class_name check is not grouped inside parenthesis. `/child::wnd[(@title=″a″ or @title=″b″) or (@class_name=″c″)]` would evaluate fine.

Validating Functions

AccessStudio allows you to validate the accuracy or completeness of the functions you configure. These include AccessProfiles, Authentication Services, Applications and Advanced Data functions like Account Data Templates. The node color changes to red when AccessStudio detects a problem with any node in a function.

There are two ways to identify the problem with a given node. You can:

❶ Use the **Validate XML Structure** function

or

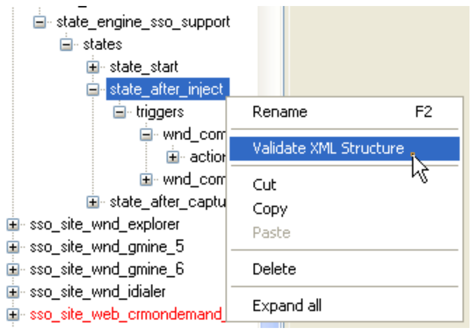
❷ Check the **Tasks** Pane.

This chapter covers the following topics:

- Using the Validate XML Structure feature
- Checking the tasks pane

Using the Validate XML Structure feature

Nodes at any level in a function can be validated for structural correctness or completeness. At any point of configuration, you can right-click a node and select **Validate XML Structure**. The validation will be carried for all sub-nodes under the selected node. An example is provided using a snapshot from a sample AccessProfile.



Notice that in the graphic, the node **state_after_inject** has been selected for validation. This means that the validation will be carried out for all triggers and, actions tied to the state, including any other associated information that has been configured like test-properties or conditions.

If you select a specific trigger under the state, the validation will be carried out only for that trigger, and actions/other options associated with that specific trigger. The highest level at which you can perform validation for a function is at the parent/ top-most level for a given function item. You cannot perform validation for a group of function items.

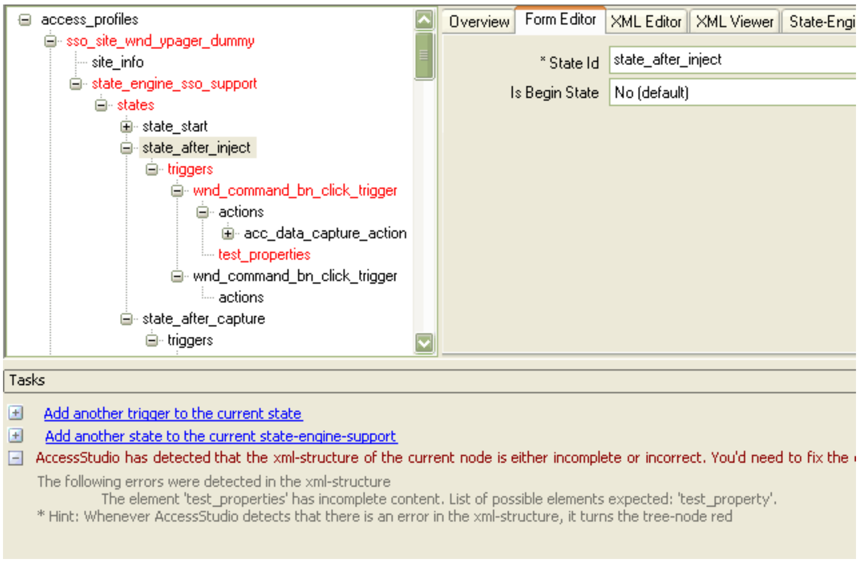
For example, you can perform a validation for individual AccessProfiles by selecting the top-most node for that specific AccessProfile, but you cannot validate the **access_profiles** node that enables you to add new AccessProfiles as it is the parent/top-most node for *all* AccessProfiles displayed in the left pane. You can also perform validation for individual authentication services, but not for a group of authentication services.

The results of the validation are displayed in the **Output** pane. The results typically indicate missing information if any, or information entered in an invalid format. You can use the results to correct the errors/fill in missing information and perform the validation again.

Checking the tasks pane

The **Tasks** pane appears at the bottom of AccessStudio by default and instantly reflects the nature of the problem with a node highlighted in red. This information will be same as that provided in the **Output** pane upon using the Validate XML Structure feature.

When you see that a particular node is red, select that node to view the associated errors in the **Tasks** pane. If a sub-node in a group is red, then the color of the nodes higher in hierarchy to the sub-nodes also turns red. An example is provided using a snapshot of a sample AccessProfile.



* State Id: state_after_inject
Is Begin State: No (default)

AccessStudio has detected that the xml-structure of the current node is either incomplete or incorrect. You'd need to fix the

The following errors were detected in the xml-structure
The element 'test_properties' has incomplete content. List of possible elements expected: 'test_property'.
* Hint: Whenever AccessStudio detects that there is an error in the xml-structure, it turns the tree-node red

Notice in the above image the following nodes are red: **sso_site_wnd_ypager_dummy, state_engine_sso_support, states, state_after_inject, triggers, wnd_command_bn_click_trigger,** and **test_properties.** The error description in the **Tasks** pane indicates that there is a problem is with the **test_properties** node.

A property must be defined under the **test_properties** node for the node and XML structure of the AccessProfile to be valid. Each parent node in the structure is also highlighted in red to enable you to trace the root of the problem. Assume that when you accessed the AccessProfiles, you saw the **sso_site_wnd_ypager_dummy** node highlighted in red. As you expand the red nodes, you identify that the problem lies in the state **state_after_inject** and can eventually identify the **test_properties** node referred to in the **Tasks** pane.

The problem indicated in the **Tasks** pane will typically also enable you to identify the solution as well. In the case of the example provided, either removing the **test_properties** node or defining a test property will solve the problem.

Testing AccessProfiles

AccessStudio can perform real-time tests on AccessProfiles. This is done using the Test function (*Test >> Start*) accessible from the **Test** menu. The results of the test are provided in the **Real-Time Logs** pane. The **Real-Time Logs** pane displays the logs of all active applications that have AccessProfiles defined for them.

For example, if you have two active applications on your computer for which AccessProfiles are defined - Windows Explorer and Internet Explorer, the Real-Time Logs pane displays two logs, once for each application.

Once you start your test, launch the applications for which you have configured AccessProfiles in AccessStudio. The test will be executed for all AccessProfiles whose corresponding applications are active on the computer. A log will be created for each one of these applications in addition to the existing logs.

For example, assume two logs were displayed in the Real-Time Logs pane based on AccessProfiles running on AccessAgent. These are for Windows Explorer and Internet Explorer. Now, assume you have configured AccessProfiles in AccessStudio for two other applications - Yahoo Messenger! and Microsoft Outlook. You then start the test and launch these applications.

The Real-Time Logs pane will now display four logs: Windows Explorer, Internet Explorer, Yahoo Messenger, and Microsoft Outlook. The first two are for AccessProfiles running on your AccessAgent. The last two are for those which you are testing using AccessStudio. You have to option to close any of these using the **Close** button at the right-hand corner of the pane.

When you start a test using AccessStudio, the Encentuate AccessAgent Wallet is temporarily cleared until the test is stopped. This means that logon automation on your computer will not work until the time the test is stopped.

This chapter covers the following topics:

- Testing AccessProfiles
- Description of the logs using a sample AccessProfile

Testing AccessProfiles

To test AccessProfiles:

❶ Access the **View** menu, select **Other Windows**, then selecting **Real-Time Logs** to open the Real-Time Logs pane.

❷ Press the **F5** key on your keyboard. You can also access the **Test** menu and select **Start F5.**

❸ Launch the applications for which you have created AccessProfiles in Access-Studio and that you want to test. For applications that are already active on your computer, you need to restart the applications before you start the test.

---

*Ensure that the application has been completely closed before you restart it. **

For applications like virus scanners you must access the 'Services' console via 'Administrative Tools' to completely shut down the application.

For applications like Windows Explorer, you must ensure that the application exe process is ended by accessing the **Windows Task Manager** *and navigating to* **Pro-cesses**. *You can then access the* **File** *menu, click* **New Task (Run)...** *and restart the application by typing the application exe name. You can alternatively start it using the* **Start** *menu.*

A log is created for each application in addition to the existing logs in the Real-Time Logs pane. The contents of these logs are explained in the coming section.

❹ You can perform the relevant actions on the application to verify if the AccessProfile is being executed as expected.

❺ Press **Shift+F5** on your keyboard to stop the test.

Description of the logs using a sample AccessProfile

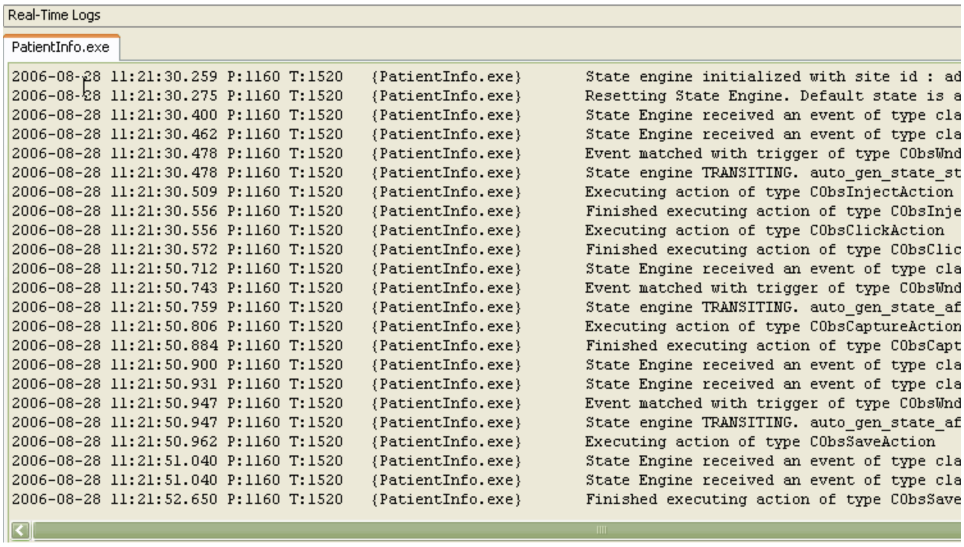
This section provides a description of the logs using an AccessProfile for a sample application.

To execute a test for any application:

❶ Run the text by pressing **F5** on your keyboard. Ensure that the Real-Time Logs pane is open.

❷ Launch the application. A new log is created for the application in the Real-Time Logs pane.

❸ Enter your user credentials in the **Username** and **Password** fields. Click **OK**. The following will be displayed in the Real-Time Logs pane.



```
2006-08-28 11:21:30.259 P:1160 T:1520   {PatientInfo.exe}       State engine initialized with site id : ad
2006-08-28 11:21:30.275 P:1160 T:1520   {PatientInfo.exe}       Resetting State Engine. Default state is a
2006-08-28 11:21:30.400 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type cla
2006-08-28 11:21:30.462 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type cla
2006-08-28 11:21:30.478 P:1160 T:1520   {PatientInfo.exe}       Event matched with trigger of type CObsWnd
2006-08-28 11:21:30.478 P:1160 T:1520   {PatientInfo.exe}       State engine TRANSITING. auto_gen_state_st
2006-08-28 11:21:30.509 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsInjectAction
2006-08-28 11:21:30.556 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsInje
2006-08-28 11:21:30.556 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsClickAction
2006-08-28 11:21:30.572 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsClic
2006-08-28 11:21:50.712 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type cla
2006-08-28 11:21:50.743 P:1160 T:1520   {PatientInfo.exe}       Event matched with trigger of type CObsWnd
2006-08-28 11:21:50.759 P:1160 T:1520   {PatientInfo.exe}       State engine TRANSITING. auto_gen_state_af
2006-08-28 11:21:50.806 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsCaptureAction
2006-08-28 11:21:50.884 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsCapt
2006-08-28 11:21:50.900 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type cla
2006-08-28 11:21:50.931 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type cla
2006-08-28 11:21:50.947 P:1160 T:1520   {PatientInfo.exe}       Event matched with trigger of type CObsWnd
2006-08-28 11:21:50.947 P:1160 T:1520   {PatientInfo.exe}       State engine TRANSITING. auto_gen_state_af
2006-08-28 11:21:50.962 P:1160 T:1520   {PatientInfo.exe}       Executing action of type CObsSaveAction
2006-08-28 11:21:51.040 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type cla
2006-08-28 11:21:51.040 P:1160 T:1520   {PatientInfo.exe}       State Engine received an event of type cla
2006-08-28 11:21:52.650 P:1160 T:1520   {PatientInfo.exe}       Finished executing action of type CObsSave
```

Log entry description

A log entry can be broken into five parts. Let's take the first log entry for the application.

```
2006-08-25 14:02:10.753 P:2372 T:2396{PatientInfo.exe}State
engine initialized with site id : sso_site_Healthcare Demo
Application
```

- 2006-08-25 14:02:10.753 -Timestamp of the log
- P:2372 -Process ID
- T:2396 - Thread ID
- {PatientInfo.exe} - The application executable for which the AccessProfile is created.
- State engine initialized with site id : sso_site_Healthcare Demo Application - State-engine information.

State-engine information log item	Description
State engine initialized with site id: adv_ap_wnd_patientinfomanager_v1	State-engine started for the AccessProfile adv_ap_wnd_patientinfomanager_v1
Resetting State Engine. Default state is auto_gen_state_start	State-engine assumed the start state defined in the AccessProfile.
Event matched with trigger of type CObsWndActivateTrigger	Window Activate trigger was fired on the activation of the **Login** window.
Executing action of type CObsInjectAction	Action to auto-fill user credentials was executed.
Executing action of type CObsClickAction	Action to click the OK button was executed.
Event matched with trigger of type CObsWndCommandTrigger	Window Command trigger was fired on the clicking of OK button. (This trigger fires when an input is received by a child window in a nested window set).
Executing action of type CObsCaptureAction	Action to capture user credentials was executed.
Event matched with trigger of type CObsWndDestroyTrigger	Window Destroy trigger was fired when logon was successful and the **Login** window disappeared. (This trigger fires when a window is destroyed).
Executing action of type CObsSaveAction	Action to save user credentials was executed.

State-engine information log item description

Downloading, Uploading, and Saving Information

AccessStudio enables you to download AccessProfiles and associated information (which includes application objects, authentication services, authentication service groups, and authentication service group links) from either IMS Server or the AccessAgent installed on your computer.

Once you create information (like a new authentication service) or modify it, you must upload it to the IMS server to be available to all AccessStudio users. AccessStudio also enables you to save the AccessProfiles and additional information you configure in a separate file.

This chapter covers the following topics:

- Dowloading information
- Uploading information
- Saving information

Dowloading information

To download AccessProfiles and associated information from the IMS Server:

1. Click the **File** menu.
2. Select **Import data from IMS [your server name]**. This will download all existing AccessProfiles and associated information on your IMS server.

AccessStudio will connect to the same IMS Server as configured for the AccessAgent on the computer. If download is successful, a dialog box appears confirming a successful download.

To download an AccessProfile from the AccessAgent installed on your computer:

❶ Click the File menu.

❷ Select Import data from local AccessAgent. This will download all existing AccessProfiles and associated information from your local AccessAgent.

Once downloaded, you can view this information, modify it, or save it in a file for future reference and use. For the changes to be applicable globally, upload this information back to IMS Server.

Uploading information

You can upload the following to IMS Server once you have created or modified them: AccessProfiles, application objects, authentication services, authentication service groups, and authentication service group links.

To upload an AccessProfile or associated information to IMS Server:

❶ Right-click the top-level node of a function.

❷ Select the Upload to IMS. For example right-click the top-most node representing an AccessProfile, like profile_1, and select Upload to IMS.

Saving information

To save AccessProfiles and associated information:

❶ Click the File menu.

❷ Select Save.

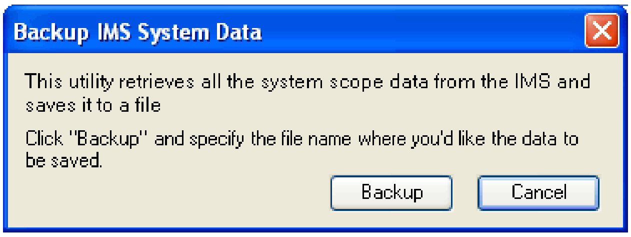
❸ Specify a name and location to save the file and click Save. All AccessProfiles and associated information will be saved in this file.

Backing up IMS Server Data

AccessStudio provides you with an option to take a backup of AccessProfiles and associated information existing on your IMS server. When you use this option, AccessStudio downloads all the information and saves them in a custom AccessStudio .eas file or XML file (depending on your preference).

To back up AccessProfiles and associated information existing on your IMS Server:

❶ Go to *Tools >> Backup System Data from IMS to File....* and the following message is displayed.


Backup IMS System Data

This utility retrieves all the system scope data from the IMS and saves it to a file

Click "Backup" and specify the file name where you'd like the data to be saved.

❷ Click the **Backup** button. You are prompted to provide information about the file name and format.

❸ Select the file format from the **Save as type** drop down list. You can either save the information in an XML file or a custom AccessStudio .eas file.

❹ Provide a file name, and choose the location to save the file in. A message is displayed confirming the success of the backup.

Frequently Asked Questions

What is Simple SSO Support?

Simple SSO Support is nothing but a Standard AccessProfile. Standard AccessProfiles are referred to as Simple SSO Support in AccessStudio. Thus when you create a new AccessProfile, you must select the Simple SSO Support option to create a Standard AccessProfile. Refer to the Standard AccessProfile section of this document for more details on Standard AccessProfiles.

What is State Engine SSO Support?

State Engine SSO Support is nothing but an Advanced AccessProfile. Advanced AccessProfiles are referred to as State Engine SSO Support in AccessStudio. Thus when you create a new AccessProfile, you must select the State Engine SSO Support option to create an Advanced AccessProfile. Refer to the Advanced AccessProfile section of this guide for more details on Advanced AccessProfiles.

What is the difference between a Standard AccessProfile and Advanced AccessProfile?

A Standard AccessProfile enables you to configure support for applications based on commonly-required information for applications. You can use the AccessProfile Generator to easily create Standard AccessProfiles. Standard AccessProfile support is sufficient for most applications.

Advanced AccessProfiles are particularly useful for workflow automation.

Refer to the Advanced concepts section of this document for detailed description of AccessProfiles.

What is the difference between Account data and Account data templates?

Account data refers to the logon information required for authentication against an authentication service. For example, user name or password. An account data template defines the nature of this information. For example, it defines that the user name field is case-insensitive and is not a secret while the password field is case-sensitive and is a secret.

Refer to the Advanced concepts section of this document for more information on account data.

Why are some of the AccessProfiles in the left pane red?

If an AccessProfile is red in color, then it indicates that the XML for that AccessProfile is not structurally correct. This means that either some required elements/attributes are missing or some extra ones are added which AccessStudio doesn't recognize.

AccessStudio not only changes the color of the sub-nodes where it finds the problem, but also that of top-level node which contains the erroneous sub-node. This means that while a complete set of nodes might be highlighted in red, the problem possibly lies in only the lowest-level sub-node. This also means that most often if you follow the trail of red nodes from the top-level node, you can easily locate the node causing the problem.

Additionally, when you select the erroneous node in the left pane, the Tasks pane describes the errors associated with that node, so that you can rectify the problem. Once you fix the error, the color of all nodes reflecting the error returns to black.

How would you define the relationship between a profile, an application and an authentication service?

A profile can be written for only one application, but can correspond to multiple authentication services.

After I launch my application, I see a pop up dialog before logon screen appears. How do I handle this app?

Generate an automated task (Select Other tasks at the task selection step) for the pop up dialog.

How do I make the same AccessProfile work for multiple versions?

UI remains the same:

You can use the Advanced settings option in the Wizard to remove variable information like version number.

UI is different and the AccessProfile stops working:

You can use the wizard to generate a new AccessProfile

My screen title contains host/IP address of the server. How do I make the same AccessProfile work for multiple hosts?

You can use the Advanced settings option in the Wizard to remove variable information like host name/IP address.

How do I know if there is already an existing AccessProfile written for an application?

You can use the *File >> Import data from IMS...* option, which will load all the available AccessProfiles in AccessStudio.

Triggers

The table below lists the pre-defined triggers in AccessStudio:

Trigger	Description
wnd_activate_trigger	Windows raises the *Activate* event on a window with a title bar when it is activated by the user or a program, using the mouse or the keyboard, or while coding. This trigger is fired when this activation of window occurs. The activation is typically characterized by the appearance of the window in the foreground with a highlighted title bar. The parameter of this trigger is the signature of the window that will be activated. Example: To trap activate on window with title say, "Messenger! signin", `<wnd_activate_trigger>` `<signature>/child::wnd[@title=""Messenger! signin"]</signature>` `</wnd_activate_trigger>` Note: *The* window activate *trigger will work only on a window with a title bar. It will only work for window(s) within the same process the agent is residing. First, the agent should be loaded into the intended process before using* activate *trigger on the window within that process. Unlike other triggers, this would not fail even if the window is not present when the trap is being set.*

Trigger	Description
wnd_command_bn_click_trigger	Windows raises the *button click* event when a button is clicked on a window. This trigger is fired when this clicking of button occurs. The parameters for this trigger are signature of the window and the button. Example: To trap the click of a button with control ID "101" within a window with the signature "/child::wnd[@title="Messenger"]", use <wnd_command_bn_click_trigger> <signature>/child::wnd[@title="Messenger"]</signature> <control> <signature>/child::wnd[@title="Messenger"]/child::wnd[@ctrl_id="101"]</signature> </control> </wnd_command_bn_click_trigger> Note: *If the window with the specified signature is not found, no event trapping would be set. To make sure the window is available, set a trap for* activate *at the top level parent window containing that window.*
web_document_complete_trigger	A *Document Complete* event takes place when a Web page has been fully loaded in the Web browser. The status bar of the browser typically indicates when a Web page has been completely downloaded and displayed. This trigger is fired when this loading completes. This trigger is usually the first trigger in the *begin* state in an AccessProfile for a Website. The parameter for this trigger includes the signature of the completed Web page. Once this trigger is executed, other triggers associated with the specific Web page (identified via the signature) are executed. Tip: *To match dynamic pages with a single URL, you can specify XPath to a unique element on a page.*

Trigger	Description
web_click_item_trigger	A Web browser raises the *click* event when an HTML element is clicked manually or automatically. This trigger fires when the HTML element is clicked. The parameter of this trigger is the signature of the HTML element which is being monitored for clicking. Example: To trap a *click* event on an image element which source is *someimage.jpg*, use <web_click_item_trigger> <signature>/child::html/child::html/child::html[@tag_name="img" and src="someimage.jpg"]</signature> </web_click_item_trigger> Note: *If the HTML element to which the* click *event has to be trapped has a Java script for that event, it means there are two possibilities when trapping the event: after or before the Java script execution. Set the* before_javascript *attribute to* 0 *to trap after the script execution.*
wnd_create_trigger	Windows raises a *create* event when a window is created. This trigger fires when this creation of a window occurs. The parameter for this trigger is the signature of the window which is being created. Example: To trap the creation of a window that would match /child::wnd[@title="Messenger"], use <wnd_create_trigger> <signature>/child::wnd[@title="Messenger"]</signature> </wnd_create_trigger> Note: *Unlike other triggers, this would not fail if the window is not present while the trap is being set.*

Trigger	Description
wnd_destroy_trigger	Windows raises an event when a window is closed. This trigger fires when the window closes. The parameter of this trigger is the signature of the window that will be closed. Example: To trap a *destroy* event on a window that says *Messenger! Messenger*, the signature would look like /child::wnd[@title="Messenger! Messenger with Voice" and @class_name="MessengerBuddyMain"] Tip: *This trigger could be used to transit the state engine in case a* mouse *event or a* key press *event cannot be captured in an application that has an owner-drawn window.*
wnd_hide_window_trigger	Windows raises the *hide window* event on a window before hiding it. This trigger is fired when event is raised. The parameter of this trigger is the signature of the window which is to be hidden Example: To trap *hide window* event on a window with text "Sign in" <wnd_hide_window_trigger> <signature>/child::wnd[@title="Sign in"]</signature> </wnd_hide_window_trigger> Note: *If the window with specified signature is not found, no event trapping would be set. To make sure the window is available, set a trap for* activate *at the top level parent window containing that window.*

Trigger	Description
wnd_is_window_found_trigger	This trigger searches for and fires when it finds the window specified in the signature parameter by polling that window. The other parameters to this trigger are the polling interval in seconds and the number of times the window should be searched for, the polling count. Example: In some cases, the wnd_is_window_found_trigger is the only choice because the wnd_activate_trigger is unable to trap the *activate* event generated by Windows. In such cases, the wnd_is_window_found is a direct substitute for the wnd_activate_trigger Note: *The use of this trigger should be avoided unless it is absolutely necessary. In general, any polling mechanism will introduce performance overheads that will spoil the user experience.*

Trigger	Description
wnd_key_down_trigger	Windows generates a *key down* event when a key or a combination of keys on the keyboard are pressed. The parameters of this trigger are the signature of the window that must be monitored for keyboard input, the key combination that should be matched, and an optional regular expression if a specified text output is to be matched. Examples: - To capture the username and password when logging in to Messenger! Messenger, specify the signature of the password field in the signature box and choose to fire the trigger on the Enter key. - For a terminal application, a sequence of key-strokes - such as a command like *change password* - can be captured using the key_down_trigger and matched against the regular expression specified in the regex parameter of the trigger. A successful match can be used to trigger the change password workflow in the state engine. The regular expression would look something like .*change.* Notes: - *This is a very powerful trigger to monitor user input and can be used in state engines that perform logon automation on terminal or mainframe applications that need heavy user interaction.* - *This trigger is very sensitive to the signature of the window that has to be monitored for keyboard input. If there are problems detecting the keyboard input on a window, try making the signature more generic or more specific. If input from a previous keyboard_input_action is to trigger a key_down_trigger, ensure that the signatures for the input action and the keydown trigger are identical.*

Trigger	Description
wnd_lbutton_down_trigger	Windows raises a *left button down* event when the left button of a mouse is clicked in an application window. This trigger fires when this clicking of the left mouse button occurs. The parameter of this trigger is the signature of the window in which the left mouse button is clicked. Example: To trap the left mouse button click the OK button, the Signature element would be / child::wnd[@title="Messenger Talk"]/ child::wnd[@class_name="Main View"]/ child::wnd[@class_name="#32770"]/ child::wnd[@class_name="Button" and @ctrl_id=1] Tip: *This trigger would normally contain a capture_action to capture the user name and password after the user clicks the OK button.*
wnd_rbutton_down_trigger	Windows raises a *right button down* event when the right button of a mouse is clicked in an application window. This trigger fires when this clicking of the right mouse button occurs. The parameter of this trigger is the signature of the window in which the right mouse button is clicked. Example: To trap the right mouse button click on an OK button, the Signature element would be / child::wnd[@title="Messenger Talk"]/ child::wnd[@class_name="Main View"]/ child::wnd[@class_name="#32770"]/ child::wnd[@class_name="Button" and @ctrl_id=1]
wnd_mdi_activate_trigger	Windows raises an *activate* events for medical applications when a child window of an application with a title bar receives focus or is selected by the user. The only parameter of this trigger is the signature of the child window that receives focus or is selected. Example: Some medical applications have child windows that can only be monitored for mdi_activate events. In such cases, the wnd_mdi_activate_trigger will have to be used instead of the wnd_activate_trigger

Trigger	Description
wnd_set_cursor_trigger	Windows raises a *set cursor* event for a window on which the cursor is active. This trigger fires when the cursor moves on the specified window. The parameter of this trigger is the signature of the window which *set cursor*" event has to be trapped. Example: To trap a "set cursor" event on window with signature /child::wnd[@title="Messenger"]/ child::wnd[@ctrl_id="102"], use <wnd_set_cursor_trigger> <signature>/child::wnd[@title="Messenger"]/ child::wnd[@ctrl_id="102"]</signature> </wnd_set_cursor_trigger>
wnd_set_focus_trigger	Windows raises the *set focus* event when the any input field on the window is clicked. This trigger is fired when clicking occurs. The parameter of this trigger is the signature of the window whose input field that is being monitored for input field click. Example: To trap a "set focus" event on window with signature / child::wnd[@title="Messenger"]/ child::wnd[@ctrl_id="102"], use <wnd_set_focus_trigger> <signature>/child::wnd[@title="Messenger"]/ child::wnd[@ctrl_id="102"]</signature>
wnd_set_text_trigger	Windows raises a *set text* event when the text of a window is modified by an external program. This trigger is fired when this modification by an external program occurs. The parameter of this trigger is the signature of the window where the text modification is to occur. Example: To trap a *set text* event on a window with signature / child::wnd[@title="Messenger"]/ child::wnd[@ctrl_id="102"], use <wnd_set_text_trigger> <signature>/child::wnd[@title="Messenger"]/ child::wnd[@ctrl_id="102"]</signature> </wnd_set_text_trigger>

Trigger	Description
wnd_show_window_trigger	Windows raises a *window show* event before a window is displayed on screen or before it is hidden (through minimization). This trigger fires before the window is displayed or hidden. The parameter of this trigger is the signature of the window which is to be displayed or hidden Example: To trap the *show* event on a window with a text that says, Sign in <wnd_show_window_trigger> <signature>/child::wnd[@title="Sign in"]</signature> </wnd_show_window_trigger> Note: *If the window with the specified signature is not found, no event trapping would be set. To make sure the window is available, set a trap to activate the top level parent window containing that window.*
wnd_window_pos_changed_trigger	Windows raises an event when the position of a window is changed on the desktop. This trigger fires when this change in window position occurs. The parameters to this trigger are the signature of the window that is to be monitored for position change and an optional flag parameter value (the flag is a member of the WINDOWPOS structure which can be obtained using Spy).
wnd_menu_click_trigger	This trigger fires when a specified menu item on the Windows application is clicked. The other parameters to this trigger are the signature of window with the menu and the path of the menu item. Example: For the trigger to fire when a menu item is clicked, specify: Signature of the window with the menu: /child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"] Menu Path: File/Login Tip: *You can specify a multi-level menu item by providing a separator '/' in the menu path. For example File/Open...*

Trigger	Description
wnd_is_console_text_found_trigger	This trigger searches for and fires when it finds the text specified in the sections parameter by polling for text in the console application screen. The other parameters to this trigger are the sections of the text to match against the screen's contents, the polling interval in seconds and the number of times the window should be searched for, the polling count. Note: *As this is a poller, use this judiciously.*
web_before_navigate_trigger	A *before navigate* event takes place when the Web browser starts navigating to a URL. This trigger is fired when this navigation to a URL begins. Example: To trap the event, user navigating out of a Web page - for example, http://www.mail.messenger.com/ - use <web_before_navigate_trigger> </web_before_navigate_trigger> Tip: *As this trigger would trap all* before navigate *events, use test property for filtering.*
web_elem_onload_trigger	The Web browser raises an *On Load* event when the loading of specific HTML elements on a Web page is complete. For example, this event and trigger are fired when the body of a HTML page finishes loading. The parameter of this trigger is the signature of the HTML element which is to complete loading for the trigger to fire. Example: To trap an *onload* event on the body element, <web_elem_onload_trigger> <signature>/child::html[@tag_name="body"]</signature>

Trigger	Description

web_key_press_trigger	A Web browser raises the *key press* event when a key is pressed for a HTML element. This trigger fires when this event is raised for the specified HTML element. The parameter of this trigger is the signature of the HTML element which is being monitored for pressing of key.
	
	Example:
	To trap a *key press* event on an image element which source is **someimage.jpg**, use
	`<web_key_press_trigger>`
	`<signature>/child::html/child::html/child::html[@tag_name="img" and src="someimage.jpg"]</signature>`
	`</web_key_press_trigger>`
	
	Note:
	If the HTML element to which the key press *event has to be trapped has a Java script for that event, it means there are two possibilities when trapping the event: after or before the Java script execution. Set the* **before_javascript** *attribute to* **0** *to trap after the script execution.*

web_browser_close_trigger	A *browser close* event takes place when the Web browser closes. This trigger is fired when this event takes place.
trm_screen_output_trigger	A terminal application raises the *terminal application screen output* event when there is text output on the application screen. This trigger fires when this text output occurs. The parameter of this trigger is a regular expression of the text that is being searched for.

mf_screen_output_ex_trigger	AccessAgent raises the *screen text change* event when there is a change in the text anywhere on a window of the process for which an AccessProfile is running. This trigger fires when this text change occurs. The parameter of this trigger is sections of the text to match against the window's contents.
	
	Example:
	To trap a "screen text change" event when a window's 10th line changes to "user name:"
	`<mf_screen_output_ex_trigger>`
	`<sections>`
	`<section>`
	`<match_text_section>`
	`<line from="10" to="10"></line>`
	`<match_text>user name:</match_text>`
	`</match_text_section>`
	`</section>`
	`</sections>`
	`</mf_screen_output_ex_trigger>`

Trigger	Description
mf_is_screen_text_found_trigger	Specifies information for the trigger to fire when there is a specific text output on a mainframe application screen. A mainframe application raises the *screen output* event when there is text out anywhere on a mainframe application window. This trigger fires when this text output occurs. The parameter of this trigger is sections of the text to match against the window's contents. Note: *As this is a poller, use this judiciously.*
hll_screen_output_trigger	The HLL framework of an HLL-compatible mainframe or terminal application generates a message when the application outputs some text to screen. This trigger is fired when this output occurs. Specific sections of text output to the screen can be matched by this trigger. The parameters of this trigger are the signature of the HLL-enabled application window and a regular expression of the text that is being searched for. Note: *This trigger will work only for an application that is HLLAPI-enabled. Common HLLAPI-enabled applications are: Attachmate Extra, IBM iSeries and Reflection.*
hll_session_start_trigger	The framework of an HLL application generates a message when an HLL-compatible mainframe session starts. This trigger fires when this session starts. The parameters to this trigger are the signature of the HLL-enabled application window and the HLLAPI short name for the application. Example: <hll_session_start_trigger> <short_name>A</short_name> <long_name></long_name> <wnd_signature>/child::wnd[@class_name="SDIMainFrame" and @title#"WS - EXTRA! X-treme"]</wnd_signature> </hll_session_start_trigger> Note: *If the short name is not configured on the EXTRA application, then this trigger will not fire.*

Trigger	Description
gen_null_trigger	This trigger fires at a zero second timeout. There are no parameters for this trigger. Example: This trigger is used to test the value of properties that were set in previous states. Notes: - *Use two gen_null_triggers in a state to control the flow in a state-engine by testing for one possible value of a boolean condition as a test_property in each trigger.* - *Do not use two gen_null_triggers in the same state without test_properties transiting to different states. The behavior of such usage cannot be predicted reliably.*
gen_sign_in_trigger	AccessAgent raises a *logon* event when an Encentuate AccessAgent user logs on. This trigger is fired when this log on occurs. The parameter of this trigger is the signature of the window on which automated logon is performed. Tip: This trigger is used to perform automated logon to an application that runs without being terminated across multiple user sessions, as soon as a user logs on to Encentuate AccessAgent.
gen_sign_out_trigger	AccessAgent raises a *logoff* event when an active Encentuate AccessAgent user logs off. This trigger is fired when this log off occurs. The parameters of this trigger are: timeout in seconds, whether logoff should be synchronized across applications, and whether logoff can happen concurrently. The default settings suffice for most cases. Example: This trigger is used to perform graceful logoff from desktop applications when a user logs off AccessAgent. The actions in this trigger can save the user's current work and log the user off or terminate the application when a logoff event is received. Tip: *Use gen_sign_out triggers to ensure that a user is logged off an application regardless of when he chooses to log off AccessAgent. There are usually multiple gen_sign_out_triggers in multiple states in a state engine.*

Trigger	Description
gen_time_out_trigger	This trigger will always fire after the time-out counter expires. The parameter to this trigger is the timeout value, in seconds. Examples: - This trigger can be used to test the value of properties that were set in previous states. - This trigger can be used to wait for an application window to stabilize in case the control IDs of the user name or password field change after window creation. - This trigger can be used as a last resort in case all the triggers in a particular state never fire, in order to prevent the state engine from freezing in that state. Tip: *Avoid using too many timeout triggers in an AccessProfile. This will make the state-engine more difficult to troubleshoot or debug. It also increases the scope for user intervention during logon automation.*
jwnd_activate_trigger	JVM raises the *Activate* event on a java window with a title bar when it is activated by the user or a program, using the mouse or the keyboard, or programmatically. This trigger is fired when this activation of java window occurs. The activation is typically characterized by the appearance of the java window in the foreground with a highlighted title bar. The parameter of this trigger is the signature of the java window that will be activated. Example: To trap activate on java window with title say, "Login", <jwnd_activate_trigger> <signature>/child::jwnd[@title="Login"]</signature> </jwnd_activate_trigger> Note: *The* java window activate *trigger will work only on a java window with a title bar. It will only work for window(s) within the same process the agent is residing. First, the agent should be loaded into the intended process before using* activate *trigger on the window within that process. Unlike other triggers, this would not fail even if the window is not present when the trap is being set. If this trigger is to be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.*

Trigger	Description
jwnd_click_trigger	Java generates the *click* event when a java window is clicked manually or automatically. This trigger fires when the java window is clicked. The parameter of this trigger is the signature of the java window which is being monitored for clicking. Example: To trap a *click* event on an java window, use <jwnd_click_item_trigger><signature>/ child::jwnd[@class_name="LoginPanel1"]/ child::jwnd[@class_name="javax.swing.JRootPane"]/ child::jwnd[@class_name="javax.swing.JLayered-Pane"]/ child::jwnd[@class_name="javax.swing.JPanel"]/ child::jwnd[@class_name="javax.swing.JPanel"]/ child::jwnd[@class_name="javax.swing.JButton" and @is_visible=1 and @title="jButton1"]</signature> </jwnd_click_item_trigger> Note: *If this trigger is to be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.*
jwnd_is_window_found_trigger	This trigger searches for and fires when it finds the java window specified in the signature parameter by polling that window. The other parameters to this trigger are the polling interval in seconds and the number of times the window should be searched for, the polling count. Note: *The use of this trigger should be avoided unless it is absolutely necessary. In general, any polling mechanism will introduce performance overheads that will spoil the user experience. If this trigger is to be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.*

Trigger	Description
jwnd_key_down_trigger	Java generates a *key down* event when a key or a combination of keys on the keyboard are pressed. The parameters of this trigger are the signature of the java window that must be monitored for key-board input and the key combination that should be matched. Example: To capture the username and password when log-ging into a bank's internet banking system, specify the signature of the password field in the signature box and choose to fire the trigger on the Enter key Notes: - *If this trigger is to be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.* - *This trigger is very sensitive to the signature of the window that has to be monitored for key-board input. If there are problems detecting the keyboard input on a window, try making the sig-nature more generic or more specific. If input from a previous keyboard_input_action is to trig-ger a key_down_trigger, ensure that the signa-tures for the input action and the keydown trigger are identical.*
java_is_jvm_available_trigger	This trigger fires when the JVM (Java Virtual Machine) is available. The other parameters to this trigger are the signature of window with the menu and the path of the menu item. Example: For the trigger to fire when a menu item is clicked, specify: Signature of the window with the menu: /child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"] Menu Path: File/Login Note: - *For java applications, this trigger should be pro-vided in the start state so that the JVM is ready before any automation or SSO starts.* - *If this trigger is to be used, the java_ses_info_item must be added to the site_info of the AccessProfile for the trigger to work.*

Actions

The table below lists the pre-defined actions in AccessStudio:

Action	Description
acc_data_inject_action	This action enables automatic insertion of user account data (credentials) on corresponding logon fields in an application window. The action first attempts to fetch user account data from the specified authentication service and eventually transfer the account data into the inject account data bag. Finally, it inserts the user account data into the respective logon fields defined in the sso_items option. Tip: *You can use different injection bags for different authentication services if you have multiple logon screens for an application.*
acc_data_capture_action	This action enables capturing of a user's credentials from the logon fields on an application window. These fields are the username and password fields. This action is defined when enabling logon automation for any application. The captured credentials are stored in an account data bag and eventually transferred to Encentuate Wallet upon the execution of an acc_data_save action. This action is executed after the Enter key is pressed for a credential field (see wnd_key_down_trigger) or after the left mouse button is clicked on an Ok or Log On button (see wnd_lbutton_down_trigger). For a Web application, this action could be contained inside a web_click_item_trigger or a web_before_navigate_trigger. Tip: *If an application is expected to run across multiple user/AccessAgent sessions, ensure that the* Empty Account Data Bag First *field value is set to* Yes *so that any cached credentials belonging to the previous user are erased.*

Action	Description
acc_data_save_action	This action saves the user's account data (credentials) from the specified account data bag to the user's Encentuate Wallet. Tip: *The action is usually used with the Account Data Capture action to save the account data from the capture bag to the wallet. So ensure that valid account data is available in the bag before the save occurs.*
data_transfer_action	This action transfers data from a source location to a destination location. To specify data transfer information, right-click the data_transfer_items node and select **Add Data-Transfer-Item**. Next, right-click the **from** and **to** nodes to specify the source and destination of the data to transfer. Example: Transfer of data from a window XPath to a property: <data_transfer_action> <data_transfer_items> <data_transfer_item> <from> <wnd_xpath> <signature>/ child::wnd[@title="Untitled - Notepad" and @class_name="Notepad"]/ child::wnd[@class_name="Edit" and @ctrl_id=15]</signature> </wnd_xpath> </from> <to> <property_store_item id="test"> </property_store_item> </to> </data_transfer_item> </data_transfer_items> </data_transfer_action> Note: - *The type of source location can be different from the destination location. For example, the source can be a field on an application screen while the destination could be an item in the account data bag.* - *The source and destination location must be specified in order for this action to be valid. You also have an option to enable data transfer only when auto-logon is enabled.*

Action	Description

keyboard_input_action	This action will send the specified keyboard input to the destination application window.
	
	Example:
	To send Ctrl+Enter to the Windows Notepad window:
	    <keyboard_input_action>
	        <keyboard_inputs>
	            <keyboard_input ctrl="1"><enter>
	                </enter>
	            </keyboard_input>
	        </keyboard_inputs>
	        <signature>/child::wnd[@title="Untitled - Notepad" and @class_name="Notepad"]/ child::wnd[@class_name="Edit" and @ctrl_id=15]</signature>
	    </keyboard_input_action>
	
	Tips:
	- *You can define multiple sets of keyboard input. To add an input set right-click the keyboard_inputs node and select* Add Key.
	
	- *There is an option to specify whether to click the target window before sending the input to make sure that the target window is on foreground.*

menu_click_action	This action clicks a specified menu item on the Windows application. Example: To click on login menu item on menu specify: Signature of the window whose menu-item is to be clicked: /child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"] Menu Path: File/Login Tip: *You can enable clicking of a multi-level menu item by providing a separator '/' in the menu path. For example File/Open...*

Action	Description
notify_signed_out_action	This action monitors and notifies whether a logoff from an application is successful. Example: To notify logoff is successful: <notify_signed_out_action></ notify_signed_out_action> Tip: *This action is useful when you have configured multiple actions to occur upon logoff completion.*
change_policy_action	This action changes the auto-fill policy of an authentication service. Example: Change the auto-fill policy of auth1 to auto_logon <change_policy_action> <policy_value>auto_logon</ policy_value> <auth_info> <direct_auth_info> <auth_id>auth1 </auth_id> </direct_auth_info> </auth_info> </change_policy_action> Tips: *The auto-fill policy can be one of the following auto_logon: Auto-fills the credentials without prompting the user, and performs auto-logon if only a single user's credentials are found. If multiple credential sets are found the user is prompted to select the credentials.* *always: Auto-fills the credentials if a single set of credentials are found, but does not auto-logon. If multiple credential sets are found the user is prompted to select the credentials.* *prompt_for_relogin: Prompt the user upon logout from the application to re-logon to the application. This prompt will display only if the auto-fill policy is set to always/automatic logon.* *ask: Prompt the user to select the credentials to be auto-filled.* *never: Disable auto-fill and auto-logon.*

Action	Description
acc_data_audit_log_action	This action enables audit logs for account data events that have occurred. Example: To provide audit logs for successful Application Logon: <acc_data_audit_log_action> <acc_data_bag_id>default_injection_bag</acc_data_bag_id> <event_code>1107296303</event_code> <return_code>0</return_code> </acc_data_audit_log_action> Tip: *You can enable audit logs for success or failure events upon application logon, and auto-fill/saving/changing of credentials.*
custom_audit_log_action	This action is used for customized audit logging.
observer_dlg_capture_action	This action prompts a dialog box that requests for a user's logon information when capturing logon credentials for applications whose fields cannot be uniquely identified. For example Owner-drawn Windows applications. Example: To prompt a dialog box for capturing user credentials: <observer_dlg_capture_action> <acc_data_bag id="default_capture_bag"></acc_data_bag> <save></save> </observer_dlg_capture_action>
observer_dlg_change_password_action	This action prompts a dialog box that requests for a user's change password information when capturing change password credentials for applications whose fields cannot be uniquely identified. For example Owner-drawn Windows applications. Example: To prompt a dialog box for capturing user's change password credentials: <observer_dlg_change_password_action><acc_data_bag id="default_capture_bag"></acc_data_bag></observer_dlg_change_password_action>

Action	Description
start_collecting_keyboard_input_action	This action starts collecting keyboard input from users for applications whose fields cannot be uniquely identified. For example Owner-drawn Windows application. Example: To start collecting keyboard input <start_collecting_keyboard_input_action> <signature>/child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"]</sig-nature> </start_collecting_keyboard_input_action> Tip: *This action is used in conjunction with Stop Col-lecting Keyboard Input action.*
stop_collecting_keyboard_input_action	This action stops collecting keyboard input from the applications for which the Start Collecting Keyboard Input action has been executed. These actions are typically used for applications whose fields cannot be uniquely identified. For example Owner-drawn Windows application. Example: To stop collecting keyboard input <stop_collecting_keyboard_input_action><sig nature>/child::wnd[@title="Form1" and @class_name="ThunderRT6FormDC"]</sig-nature></ stop_collecting_keyboard_input_action> Tip: *This action is used in conjunction with* Start Col-lecting Keyboard Input *action.*

Action	Description
web_scr_login_action	Action to enable logon for an SCR-enabled Web application. The action initiates the pass-word-less Session Challenge Response (SCR) authentication mechanism between AccessAgent and the Web application. Example: To enable SCR/password-less logon for a Web application that supports SCR, create a web_scr_login_action inside a web_document_complete_trigger. Notes: - *The site_info section of the sso_site containing this web_scr_login_action should have an SCR-enabled application.* - *The web application on which the web_scr_login_action is used should be SCR enabled.* - *This action will have no effect on a web application that is not configured for SCR.*
web_scr_registration_action	This action registers with an SCR-enabled Website. Example: To enable SCR/no-password registration for a Web application that supports SCR, create a web_scr_login_action after capturing the user's credentials for the application. Notes: - *The contents in the account data bag will be used to register with the given Web site.* - *This action must be used together with or after a web_doc_complete trigger. The user's credentials must be validated so that SCR registration succeeds. Invalid credentials will result in a failed SCR registration.*
web_set_enabled_action	This action is used for enabling or disabling a element on Web pages. Note: *The element should support IHTMLElement3 interface.*
web_set_visibility_action	This action is used for showing or hiding a element on Web pages. Note: *The element should support IHTMLElement2 interface.*

Action	Description
wnd_set_check_box_state_action	This action sets the state of a check-box on a Windows application by marking it selected or deselected. **Example:** To deselect a given check-box: `<wnd_set_check_box_state_action> <signature>/child::wnd[@title="Login" and @class_name="#32770"]/ child::wnd[@class_name="Button" and @ctrl_id=1005]</signature> <set_checked>0</set_checked> </wnd_set_check_box_state_action>` **Tip:** *There is an advanced option to specify to perform the action only if the auto logon policy is enabled.*
wnd_shortcut_action	This action will send a keyboard shortcut shortcut to a specified Windows application window. **Example:** Send the shortcut Ctrl+A to the Notepad window `<wnd_shortcut_action> <signature>/child::wnd[@title="Untitled - Notepad" and @class_name="Notepad"]</signature> <shortcut_string ctrl="1">s</shortcut_string> </wnd_shortcut_action>` **Tip:** *You can send shortcuts with multiple assistant keys such as Ctrl+Alt+A.*
wnd_close_window_action	This action closes a specified window. **Example:** Close the notepad window `<wnd_close_window_action> <signature>/child::wnd[@title="Untitled - Notepad" and @class_name="Notepad"]</signature> </wnd_close_window_action>` **Tip:** *Using the advanced options, you can specify if you want the action to be performed only if the automatic logon policy is available.*
wnd_set_visibility_action	This action is used for showing or hiding a windows control.)

Action	Description
wnd_set_enabled_action	This action is used for enabling or disabling a windows control.
menu_set_enabled_action	This action is used for enabling or disabling a windows menu item.
kill_process_action	This action ends the specified process Example: To end Internet Explorer <kill_process_action> <process_name>iexplore.exe</ process_name> </kill_process_action>
run_script_action	This action supports execution of a custom script. The languages currently supported are VBScript and JavaScript. Example: To execute script to display current date and time: <run_script_action> <script language="VBScript">docu- ment.write("Date : " & date()) docu- ment.write("Time : " & time())></script> </run_script_action> Tip: *You can specify the script type in the* Advanced Options *section.*
jwnd_click_action	This action is used for clicking a Java component. Note: *The click is done by sending mouse input to the given Java component.*
wnd_start_installing_bho_action	The action is used to hook into Windows applications, which host Internet Explorer Browser Controls on their own to enable Web page browsing. The hook will enable us listening to Web page events and performing actions on the Web page. Notes: - *If no signature is specified, all windows created within the application with class name* Internet Explorer_Server *will be hooked.* - *The window should be of class* Internet Explorer_Server.

Action	Description
wnd_stop_installing_bho_action	The action is used to stop the hook into Windows applications, which host Internet Explorer Browser Controls on their own to enable Web page browsing. Note: *All the existing hooks will remain until the hooked window is closed.*
wnd_click_action	The action will click the specified window at the given position. The position must be defined in terms of coordinates with respect to the top left corner of that window. Example: Click the notepad editor window at position 1,1 and do the click by sending user input <wnd_click_action> <signature>/child::wnd[@title="Untitled - Notepad" and @class_name="Notepad"]/child::wnd[@class_name="Edit" and @ctrl_id=15]</signature> <position> <x>1</x> <y>1</y> </position> <do_simple_click>1</do_simple_click> </wnd_click_action> Notes: - *You can specify if you want to bring the window to the foreground before clicking.* - *There are two types of click simulations - click by simulating the mouse actions (up or down) and click by providing input directly.*
web_click_action	This action clicks a specified element on a Web page. Example: To enable clicking of an OK button of a Web site, capture the signature of the OK button using the Finder tool. Specify this action for a web_document_complete_trigger that fires when the page with the OK button has fully loaded. Tip: *Using this action you can enable clicking of any Web element such as a hyperlink, button, or image link.*

Glossary

AccessProfiles

Short, structured XML files that enable single sign-on/sign-off automation for applications. AccessStudio can be used to generate AccessProfiles.

AccessProfile and associated data

AccessProfile and associated data indicates information that is necessary for an AccessProfile to be active, but is configured separately using AccessStudio before being used to define an AccessProfile. This information includes authentication service, application objects, authentication service groups, and authentication service group links.

AccessProfile generator

AccessProfile Generator is a wizard that provides a step-by-step guide to easily creating AccessProfiles.

AccessStudio

The interface used to create AccessProfiles required to support end-point automation, including single sign-on, single sign-off, and customizable audit tracking.

account data

Account data is the logon information required for verification against an authentication service. Most of the time, account data refers to the user name, password and the authentication service for which the logon information is stored.

account data templates

The account data in AccessStudio is stored in a specific format known as Account Data Templates. These provide information about the data to be captured, such as which fields are key fields, which are case-sensitive, and which are to be kept secret. See also *Account Data*.

action

An action is an act that can be performed in response to a trigger. For example automatic filling of user name and password details as soon as a sign-on window appears. See also *Trigger.*

ActiveCode

ActiveCodes are short-lived authentication codes that are controlled by the Encentuate TCI system. There are two types of ActiveCodes: random ActiveCodes and predictive ActiveCodes.

The generation of ActiveCodes can be triggered in one of two ways: time-based (e.g. every minute or every day) or event-based (e.g. pressing a button).

Combined with alternative channels or devices, ActiveCodes provide effective second-factor authentication.

advanced AccessProfiles

Advanced AccessProfiles are a set of instructions to automate applications where logon is based on various conditions. Such AccessProfiles are created for applications with complex logon situations such as logon using multiple screens, or verification of conditions before automatic logon.

application

An application is a software product that provides a particular function. Examples include customer relationship management systems and supply-chain management systems.

application group

An application group is a set of application that share the same directory. In other words, a user can logon to any of the applications in the application group using the same user name.

application object

An application object is a reference to the entity that represents the group to which the AccessProfile belongs.

application policy

Application policy is a collection of attributes governing access to applications. Application policies could include (but are not limited to):

- Password policies govern frequency of password change and strength of passwords.
- Audit policies determine if audit trails should be kept.
- Management policies determine the degree of control the user has over password auto-fill. This replaces personal versus enterprise applications.

Privacy policies define when and what type of information is captured and backed-up in the Encentuate Wallet.

auto-capture

The auto-capture function allows the system to remember user credentials (such as user names and passwords) for different applications. These credentials are captured as they are being used for the first time, and then stored and secured in the Encentuate Wallet for future use.

auto-inject

The auto-inject function allows the system to automatically enter user credentials (such as user names and passwords) for different applications, via logon automation.

authentication service

Authentication service is the entity that validates the logon information for the application. In AccessStudio it is a reference to the entity that validates the sign-on information for the application.

authentication service ID

The authentication service ID identifies the authentication service that this screen belongs to. Account data saved under a particular authentication service will be retrieved and auto-filled for the logon screen being defined. Correspondingly, account data captured from the logon screen defined will be saved under this authentication service.

control

A control is any field on a screen. For example user name text box or an OK button on a Web page.

credentials

Credentials refer to user names, passwords, certificates and any other information that is required for authentication. An authentication factor can serve as a credential. In Encentuate TCI, credentials are stored and secured in the Encentuate Wallet.

directory

A directory is a structured repository of information on people and resources within an organization, facilitating management and communication.

directory service

A directory of names, profile information and machine addresses of every user and resource on the network. It is used to manage user accounts and network permissions. When sent a user name, it returns the attributes of that individual, which may include a telephone number as well as an e-mail address. Directory services use highly specialized databases that are typically hierarchical in design and provide fast lookups.

DLL

Dynamic Link Library

DNS

Domain Name Service

Encentuate Wallet Password

The Encentuate Wallet Password is a password used to secure access to an Encentuate Wallet.

Enterprise Access Security (EAS)

A technology that enables enterprises to simplify, strengthen and track access to digital assets and physical infrastructure.

Simplifying access means time-to-information, user productivity, and convenience. Strengthening access allows stronger security and better risk management. Tracking access enables compliance.

EAS solutions are a new generation of identity management security products that reflect the convergence of logon/log-off automation, authentication management, centralized user access administration, and the unification of logical (information) and physical (building) access control systems.

Enterprise Single Sign-On (ESSO)

A mechanism that allows users to log on to all applications deployed in the enterprise by entering a user ID and other credentials (such as a password). Many ESSO products use sign-on automation technologies to achieve SSO—users logon to the sign-on automation system and the system logs on the user to all other applications.

identity wallet

A secured data store for a user's access credentials and related information (including user IDs, passwords, certificates, encryption keys). The Wallet is an identity wallet.

fortified password

A fortified password is an application password that is automatically changed without human intervention. In Encentuate TCI, passwords might be fortified with Encentuate ActiveCodes.

info-controls

Information controls are fields on a screen from which AccessAgent captures or auto-fills credential information.

HLLAPI

HLLAPI is an abbreviation for High Level Language API, which is a standard to access legacy information and applications housed on IBM and Unisys mainframes, AS/400, UNIX/VMS, etc. Examples of applications that provide HLLAPI are Attachmate Extra, Reflection, and IBM ISeries. Terminal Emulators are typically used to connect to these legacy applications.

IMS Server

An integrated management system that provides a central point of secure access administration for an enterprise. It enables centralized management of user identities, AccessProfiles, authentication policies. provides loss management, certificate management and audit management for the enterprise.

logon automation

Logon automation is technology that works with application user interfaces to automate the logon process for users. Many ESSO products use logon automation technologies to achieve SSO—users log onto the logon automation mechanism and the logon automation system takes over from there to log the user onto all other applications.

server locator

Server locators are used to group a related set of Web applications that require authentication by the same authentication service. In AccessStudio server locators are used to identify the authentication service with which an application screen is associated.

single sign-on (SSO)

A capability that allows a user to enter a user ID and password to access multiple applications.

signature

Signature is unique identification information for any application, window, or field.

sign-on information

Sign-on information is information required to provide access to users to any secure application. This information can include user names, passwords, domain information and certificates.

simple SSO support

Simple SSO Support refers to Standard AccessProfiles in AccessStudio. Refer to *standard AccessProfiles*.

SSO-items

SSO-items are fields on a screen from which AccessAgent captures or auto-fills credential information. See also *info-items*.

standard AccessProfiles

Standard AccessProfiles are a set of instructions to automate applications where all logon information is contained within a single screen.

states

State Engine SSO Support refers to Advanced AccessProfiles in AccessStudio. Refer to *Advanced AccessProfiles*.

trigger

Triggers are events that cause transitions between states in a states engine, for example, the loading of a Web page or the appearance of window on the desktop.

user name (user ID)

A unique identifier that differentiates the user from all other users in the system.

Wallet

An identity wallet that stores a user's access credentials and related information (including user IDs, passwords, certificates, encryption keys), each acting as the user's personal meta-directory.

Index